

Specification for

Powerline Intelligent Metering Evolution



Prepared by the PRIME Alliance Technical Working Group

This document is an approved specification. As such, it is subject to change. Prior to the full or partial adoption of this document by any standards development organization, permission must be obtained from the PRIME Alliance.

Abstract:

This is a complete specification for a new OFDM-based power line communication system for the provision of all kinds of Smart Grid services over electricity distribution networks. The specification also provides RF communication prescriptions based on some of the SUN FSK requirements present in the consolidated IEEE 802.15.4 standard. Both PHY and MAC layers according to IEEE conventions, plus a Convergence layer, are described in the Specification.

Content Table

Content Table	2
List of Figures	10
List of Tables	18
1 Introduction	25
1.1 Scope	25
1.2 Overview.....	25
1.3 Normative references.....	25
1.4 Document conventions	28
1.5 Definitions	29
1.6 Abbreviations and Acronyms.....	30
2 General Description	35
2.1 Introduction.....	35
2.2 General description of the architecture	35
3 Physical layer	36
3.1 Introduction.....	36
3.2 Overview.....	37
3.2.1 General	37
3.2.2 Note about backwards compatibility with PRIME v1.3.6	38
3.3 PLC PHY.....	38
3.3.1 PHY parameters	38
3.3.2 Preamble, header and payload structure.....	40
3.3.3 Convolutional encoder	50
3.3.4 Scrambler.....	51
3.3.5 Repeater	51
3.3.6 Interleaver	52
3.3.7 Modulation	53

3.3.8	Electrical specification of the transmitter	55
3.4	RF PHY	57
3.4.1	PRIME profile of SUN FSK PHY	57
3.5	PHY service specification	59
3.5.1	General	59
3.5.2	PHY Data plane primitives	60
3.5.3	PHY Control plane primitives	64
3.5.4	PHY Management primitives	73
4	MAC layer	81
4.1	Overview	81
4.2	Addressing	83
4.2.1	General	83
4.2.2	Example of address resolution	85
4.2.3	Broadcast and multicast addressing	86
4.3	MAC functional description	86
4.3.1	Service Node start-up	86
4.3.2	Starting and maintaining Subnetworks	89
4.3.3	Channel Access	90
4.3.4	Tracking switches and peers	98
4.3.5	Switching	101
4.3.6	Direct connections	103
4.3.7	Packet aggregation	108
4.3.8	Security	109
4.4	MAC PDU format	120
4.4.1	General	120
4.4.2	Generic MAC PDU	121
4.4.3	Promotion Needed PDU	156

4.4.4	Beacon PDU	158
4.5	MAC Service Access Point.....	163
4.5.1	General	163
4.5.2	Service Node and Base Node signalling primitives.....	164
4.5.3	Base Node signalling primitives	173
4.5.4	Service and Base Nodes data primitives.....	174
4.5.5	MAC Layer Management Entity SAPs	175
4.6	MAC procedures	187
4.6.1	Registration process	187
4.6.2	Unregistration process	190
4.6.3	Promotion process.....	191
4.6.4	Demotion process.....	199
4.6.5	Keep-Alive process.....	201
4.6.6	Connection establishment.....	206
4.6.7	Multicast group management	208
4.6.8	Robustness Management	215
4.6.9	Channel allocation	217
4.6.10	Programmed Configuration Change	219
4.6.11	Channel hopping.....	219
4.6.12	Network Attenuation Configuration.....	223
4.7	Automatic Repeat Request (ARQ)	224
4.7.1	General	224
4.7.2	Initial negotiation	224
4.7.3	ARQ mechanism	225
4.7.4	ARQ packets switching	228
4.8	Time Reference.....	228
4.9	Backward Compatibility with PRIME 1.3.6	229

4.9.1	Frame Structure and Channel Access	229
4.9.2	Switching	231
4.9.3	PDU Frame Formats.....	231
5	Convergence layer	232
5.1	Overview.....	232
5.2	Common Part Convergence Sublayer (CPCS)	232
5.2.1	General	232
5.2.2	Segmentation and Reassembly (SAR).....	232
5.3	NULL Service-Specific Convergence Sublayer (NULL SSCS)	234
5.3.1	Overview.....	234
5.3.2	Primitives	234
5.4	IPv4 Service-Specific Convergence Sublayer (IPv4 SSCS)	235
5.4.1	Overview.....	235
5.4.2	Address resolution.....	236
5.4.3	IPv4 packet transfer.....	238
5.4.4	Segmentation and reassembly	239
5.4.5	Header compression.....	239
5.4.6	Quality of Service mapping.....	239
5.4.7	Packet formats and connection data.....	240
5.4.8	Service Access Point	246
5.5	IEC 61334-4-32 Service-Specific Convergence Sublayer (IEC 61334-4-32 SSCS)	250
5.5.1	General	250
5.5.2	Overview.....	250
5.5.3	Address allocation and connection establishment	251
5.5.4	Connection establishment data format	252
5.5.5	Packet format	253
5.5.6	Service Access Point	253

5.6	IPv6 Service-Specific Convergence Sublayer (IPv6 SSCS)	255
5.6.1	Overview	255
5.6.2	IPv6 Convergence layer	256
5.6.3	IPv6 Address Configuration	257
5.6.4	IPv6 Packet Transfer	259
5.6.5	Segmentation and reassembly	262
5.6.6	Compression	262
5.6.7	Quality of Service Mapping	263
5.6.8	Packet formats and connection data.....	263
5.6.9	Service access point.....	268
6	Management plane	274
6.1	Introduction.....	274
6.2	Node management.....	275
6.2.1	General	275
6.2.2	PHY PIB attributes.....	275
6.2.3	MAC PIB attributes	279
6.2.4	Application PIB attributes.....	313
6.3	Firmware upgrade	315
6.3.1	General	315
6.3.2	Requirements and features	315
6.3.3	General Description.....	315
6.3.4	Firmware upgrade PIB attributes	318
6.3.5	State machine.....	318
6.3.6	Examples.....	333
6.4	Management interface description.....	334
6.4.1	General	334
6.4.2	Payload format of management information	336

6.4.3	NULL SSCS communication profile	340
6.4.4	Serial communication profile	340
6.4.5	TCP communication profile	341
6.5	List of mandatory PIB attributes.....	341
6.5.1	General	341
6.5.2	Mandatory PIB attributes common to all device types.....	341
6.5.3	Mandatory Base Node attributes	343
6.5.4	Mandatory Service Node attributes	343
Annex A (informative) Examples of CRC.....		345
Annex B (normative) EVM calculation.....		348
Annex C (informative) Interleaving matrixes ($N_{CH} = 1$).....		349
Annex D (normative) MAC layer constants		351
Annex E (normative) Convergence layer constants		353
Annex F (normative) Profiles		354
F.1	Smart Metering Profile	354
Annex G (informative) List of frequencies used		356
Annex H (informative) Informative		365
H.1	Data exchange between to IP communication peers.....	365
H.2	Joining a multicast group.....	367
Annex I (informative) ARQ algorithm		369
Annex J (normative) PHY backwards compatibility mechanism with PRIME v1.3.6		370
Annex K (normative) MAC Backward Compatibility PDUs and Procedures		374
K.1	MAC PDU format	374
K.1.1	Generic MAC PDU	374
K.1.2	Compatibility Beacon PDU (CBCN)	385
K.2	MAC procedures.....	388
K.2.1	Registration process	388

K.2.2	Unregistering process	390
K.2.3	Promotion process.....	390
K.2.4	Demotion process.....	393
K.2.5	Keep-Alive process.....	394
K.2.6	Connection management	395
K.2.7	Multicast group management	395
K.2.8	Robustness Management	395
K.2.9	Channel allocation and deallocation	395
Annex L (informative) Type A, Type B PHY frames and Robust modes.....		396
Annex M (informative) Channel Hopping examples		397
M.1	Channel sequence generation examples.....	397
M.1.1	First example	397
M.1.2	Second example.....	399
M.2	Channel transitions with different frame and CFP/SCP lengths: examples	400
Annex N (normative) Management Information Base for PRIME Nodes.....		402
N.1	Basic requirements for Base Nodes.....	402
N.1.1	Instantaneous Objects.....	402
N.1.2	Periodic Objects.....	405
N.2	Advanced Requirements for Base Nodes	406
N.2.1	Topology	406
N.2.2	Traffic Sniffer and Topology Logging.....	408
N.2.2.1	Extension for Multi-PHY Traffic Sniffer and Topology Logging.....	409
Annex O (informative) Traffic Sniffer and Topology Logging Protocol Definition.....		410
O.1	MAC_TX and MAC_RX	410
O.2	Topology	413
O.3	TopologyMP.....	413
Annex P (normative) Node and network coverage calculation.....		415

P.1	Calculation of EntryCoverage	415
P.1.1	When using PRIME 1.3.6 Alive.....	415
P.1.2	When using PRIME 1.4 Alive:.....	415
P.2	Calculation of NetworkCoverage.....	416
Annex Q (informative) TCP transport layer for DLMS: extension for optimal multiplexing 4-32 connections		417
Q.1	Introduction.....	417
Q.2	DLMS/TCP	417
Q.2.1	TCP configuration	417
Q.2.2	DLMS over TCP standard protocol.....	417
Q.2.3	Reserved ports.....	418
Q.2.4	DLMS/TCP standard compatibility.....	418
Q.3	DLMS/TCP custom messages.....	418
Q.3.1	Functionality of the custom messages	418
Q.3.2	Addressing of the custom messages	418
Q.3.3	Custom message definition	419
Q.4	Use cases	423
Q.4.1	Service Node registration	423
Q.4.2	DLMS communication.....	425
Q.4.3	Service Node deletion and deregistration.....	427
Q.4.4	Phase detection	429
List of authors (by alphabetical order).....		430

List of Figures

Figure 1 - Reference model of protocol layers used in the PRIME specification	35
Figure 2 - Overview of PPDU processing	37
Figure 3 - PHY frame of Type A format.....	37
Figure 4 - PHY frame of Type B format.....	38
Figure 5 - Example of the preamble structure when three channels are used.....	42
Figure 6 - Preamble Type B structure	42
Figure 7 - Example of each of the preamble symbol structure when three channels are used	43
Figure 8 - Pilot and data subcarrier frequency allocation inside the header (eight active channels case).....	44
Figure 9 - LFSR for use in Pilot sequence generation	44
Figure 10 - PHY frame of Type A, pilot and data subcarrier allocation (eight active channels case).....	45
Figure 11 - PHY frame of Type B, pilot and data subcarrier allocation (eight active channels case)	46
Figure 12 - PRIME PPDU of Type A: header and payload (bits transmitted before encoding)	47
Figure 13 - PRIME PPDU of Type B: header and payload (bits transmitted before encoding).....	49
Figure 14 - Convolutional encoder	50
Figure 15 - LFSR for use in the scrambler block.....	51
Figure 16 - Example of repeater block using a shift value = 2	51
Figure 17 - DBPSK, DQPSK and D8PSK mapping	53
Figure 18 - Subcarrier Mapping	54
Figure 19 – Measurement set up (single-phase)	55
Figure 20 – Measurement set up (three-phase)	56
Figure 21 – Artificial mains network.....	56
Figure 22 – EVM meter (block diagram).....	57
Figure 23 – Overview of PHY primitives	60
Figure 24 – Overview of PHY Control Plane Primitives	65
Figure 25 - Service Node states	81
Figure 26 - Subnetwork example.....	83

Figure 27 - Addressing Structure	84
Figure 28 – Example of address resolution: phase 1.....	85
Figure 29 – Example of address resolution: phase 2.....	85
Figure 30 – Example of address resolution: phase 3.....	85
Figure 31 – Example of address resolution: phase 4.....	86
Figure 32 - Band Scanning algorithm.....	88
Figure 33 - RF Channel Scanning Algorithm	89
Figure 34 – Structure of a MAC Frame	91
Figure 35 - Flow chart for PL CSMA-CA algorithm.....	95
Figure 36 - Flow chart for SUN FSK PHY CSMA-CA algorithm.....	97
Figure 37 –Switching tables examples.....	99
Figure 38 – Filling example for the switching table for Switch of Level 0.	100
Figure 39 – Directed Connection to an unknown Service Node	104
Figure 40 - Example of direct connection: connection establishment to a known Service Node.....	106
Figure 41 - Release of a direct connection	107
Figure 42 – Nonce structure	111
Figure 43 – Counter handling in ALV and request/response messages	113
Figure 44 – Counter handling in direct connections and request messages.....	114
Figure 45 – REG Nonce structure	115
Figure 46 – Key derivation hierarchy.....	115
Figure 47 – Security profile 1 and 2 encryption algorithm (authentication only).....	119
Figure 48 – Security profile 1 and 2 encryption algorithm (authentication and encryption)	119
Figure 49 – Generic MAC PDU format.....	121
Figure 50 - Generic MAC header	121
Figure 51 - Packet structure	122
Figure 52 – Packet Header.....	122
Figure 53 - PKT.CID structure.....	123

Figure 54 – Security subheader	125
Figure 55 – Two transactions without requiring retransmits.....	128
Figure 56 - Transaction with packet loss requiring retransmits.....	129
Figure 57 – Duplicate packet detection and elimination	129
Figure 58 - REG control packet structure	136
Figure 59 - CON control packet structure.....	137
Figure 60 - PRO_REQ_S control packet structure	140
Figure 61 - PRO control packet structure	140
Figure 62 - PRO_REQ_S Multi PHY control packet structure.....	144
Figure 63- PRO_REQ_B Multi PHY control packet structure	144
Figure 64 - FRA control packet structure.....	145
Figure 65 - CFP control packet structure	146
Figure 66 - ALV_RSP_S / ALV_REQ_B Control packet structure	147
Figure 67 - ALV_ACK_B/ALV_ACK_S control packet structure	148
Figure 68 - ALV_RSP_ACK Control packet structure.....	148
Figure 69 - MUL control packet structure	151
Figure 70 – SEC control packet structure	154
Figure 71 – PCC – PHY CH control packet structure	155
Figure 72 – PCC PLC ATT control packet structure	155
Figure 73 - Promotion Need MAC PDU	157
Figure 74 – Beacon PDU structure.....	159
Figure 75 – Establishment of a Connection.....	163
Figure 76 – Failed establishment of a Connection	163
Figure 77 – Release of a Connection	163
Figure 78- Transfer of Data.....	163
Figure 79 – Registration process accepted.....	189
Figure 80 – Registration process rejected	189

Figure 81 – Unregistration process initiated by a Terminal Node.....	190
Figure 82 – Unregistration process initiated by the Base Node.....	191
Figure 83 – Promotion process initiated by a Service Node	192
Figure 84 – Promotion process rejected by the Base Node	192
Figure 85 – Promotion process initiated by the Base Node	192
Figure 86 – Promotion process rejected by a Service Node.....	193
Figure 87 – BCN modulation change request initiated by the Switch.....	193
Figure 88 – BCN modulation change request initiated by the Switch and rejected by the Base Node.	194
Figure 89 - BCN modulation change request initiated by the Base Node.....	194
Figure 90 - BCN modulation change request initiated by the Base Node and rejected by the Switch	195
Figure 91 – Multi-PHY promotion process initiated by Service Node with the most relevant values	197
Figure 92 - Multi-PHY promotion process rejected by Base Node with the most relevant values.....	197
Figure 93 – Multi-PHY promotion process initiated by Base Node with the most relevant values	198
Figure 94 – Multi-PHY promotion process rejected by Service Node with the most relevant values	198
Figure 95 – Two promotion processes initiated by Service Node, one with PRO_REQ_S the other with PRO_REQ_S_MultiPHY	199
Figure 96 – Demotion process initiated by a Service Node. From Switch to Terminal.	200
Figure 97 – Demotion process initiated by the Base Node. From Switch to Terminal.	200
Figure 98 - Demotion process initiated by a Service Node. From Switch in two media to Switch in one medium.	201
Figure 99 - Demotion process initiated by a Base Node. From Switch in two media to Switch in one medium.	201
Figure 100 - Successful ALV procedure	203
Figure 101 - Failed ALV procedure	203
Figure 102 - Failed ALV procedure (uplink)	204
Figure 103 - ALV.REP_U Example (ALV.MIN_LEVEL = 2)	205
Figure 104 - ALV.REP_D Example (ALV.MIN_LEVEL = 2).....	205
Figure 105 – Connection establishment initiated by a Service Node.....	207
Figure 106 – Connection establishment rejected by the Base Node	207

Figure 107 – Connection establishment initiated by the Base Node	207
Figure 108 – Connection establishment rejected by a Service Node	207
Figure 109 – Disconnection initiated by a Service Node	208
Figure 110 – Disconnection initiated by the Base Node	208
Figure 111 – Successful group join initiated by Base Node	209
Figure 112 – Failed group join initiated by Base Node	210
Figure 113 – Successful group join initiated by Service Node	211
Figure 114 – Failed group join initiated by Service Node	212
Figure 115 – Leave initiated by the Base Node	213
Figure 116 – Leave initiated by the Service Node	213
Figure 117 - Multicast Switching Tracking for Group Join	214
Figure 118 - Multicast Switching Tracking for Group Leave	215
Figure 119 – Successful allocation of CFP period	218
Figure 120 – Successful channel de-allocation sequence	218
Figure 121 – Deallocation of channel to one device results in the change of CFP allocated to another	219
Figure 122 – CHANNELS shuffle algorithm	221
Figure 123 - ARQ subheader only with the packet id	225
Figure 124 - ARQ subheader with ARQ.INFO	225
Figure 125 - ARQ.ACK byte fields	225
Figure 126 - ARQ.WIN byte fields	225
Figure 127 - ARQ.NACK byte fields	225
Figure 128 - Example of an ARQ subheader with all the fields present	226
Figure 129 - Stop and wait ARQ subheader with only packet ID	227
Figure 130 - Stop and wait ARQ subheader with an ACK	228
Figure 131 - Stop and wait ARQ subheader without data and with an ACK	228
Figure 132 - CBCN Frame format for backwards compatibility mode	230
Figure 133 - Robust beacon allocation in 1.4 BC	230

Figure 134 - Structure of the Convergence layer	232
Figure 135 – Segmentation and Reassembly Headers	233
Figure 136 - IPv4 SSCS connection example.....	236
Figure 137 - IPv6 SSCS connection example.....	257
Figure 138 - Management plane. Introduction.	274
Figure 139 – Restarting de nodes and running the new firmware.....	317
Figure 140 – Signed firmware diagram	317
Figure 141 - Firmware Upgrade mechanism, state diagram	321
Figure 142 - Init Service Node and complete FW image	333
Figure 143 - Execute upgrade and confirm/reject new FW version.....	334
Figure 144 – Management data frame.....	335
Figure 145 – Get PIB Attribute query. Payload.....	336
Figure 146 - Get PIB Attribute response. Payload	336
Figure 147 – Set PIB attribute. Aggregated payload	337
Figure 148 – Set PIB Attribute. ACTION PIB attributes.....	337
Figure 149 – Enhanced PIB query format.....	338
Figure 150 – Iterator short format	338
Figure 151 – Iterator long format.....	338
Figure 152 – Enhanced PIB response format	339
Figure 153 – Data encapsulations for management messages.....	340
Figure 154 – Example of CRC in MAC Header	346
Figure 155 – Example of CRC in Generic PDU.....	346
Figure 156 – Example of CRC in Promotion Needed PDU	347
Figure 157 – Example of CRC in PRIME 1.4 Beacon PDU.....	347
Figure 158 – Example of CRC in PRIME 1.4 Compatibility Beacon PDU	347
Figure 159 - MSC of IPv4 SSCS services	366
Figure 160 - Joining MS.....	367

Figure 161 - MSC 432 SSCS services	368
Figure 162 - Backwards Compatible PHY frame	370
Figure 163 - BC frame predefined content.....	371
Figure 164 - PHY BC frame detected by v1.3.6 devices.....	372
Figure 165 - BC PHY frame detected by v1.4 devices.....	372
Figure 166 - v1.3.6 frame received by a v1.4 node	373
Figure 167 - BC PHY frame in noisy environment.....	373
Figure 168 – Compatibility Packet Header	374
Figure 169 - CREG control packet structure	376
Figure 170 - CPRO_REQ_S control packet structure	380
Figure 171 - CPRO control packet structure	380
Figure 172 - CBSI control packet structure.....	383
Figure 173 - CFRA control packet structure.....	384
Figure 174 - CALV Control packet structure	385
Figure 175 – Beacon PDU structure	386
Figure 176 – Registration process accepted.....	389
Figure 177 – Registration process rejected	389
Figure 178 – Promotion process initiated by a Service Node.....	391
Figure 179 – Promotion process rejected by the Base Node	391
Figure 180 – Promotion process initiated by the Base Node	392
Figure 181 – Promotion process rejected by a Service Node.....	392
Figure 182 - Double switching BSI message exchange	393
Figure 183 – Demotion process initiated by a Service Node.....	394
Figure 184 – Demotion process initiated by the Base Node	394
Figure 185 - Example with FrameLength = 276 symbols.....	400
Figure 186 - Example with FrameLength = 552 symbols.....	401
Figure 187 – Example of Service Node registration	424

Figure 188 – Example of DLMS communication.....	426
Figure 189 – Example of Service Node deletion and deregistration	428
Figure 190 – Example of Phase detection request	429

List of Tables

Table 1 - Frequency and Timing Parameters of the PRIME PHY	38
Table 2 - PHY Payload Parameters	39
Table 3 – PHY Header Parameters.....	40
Table 4 - Roll-off region length for all N_{CH} values	41
Table 5 - Roll-off region length for all N_{CH} values	43
Table 6 - Length in bits of MPDU1 and PAD_H fields in the PHY frame header of Type A for all possible values of N_{CH}	48
Table 7 - Length in bits of MPDU1 and PAD_H fields in the PHY frame header of Type B for all possible values of N_{CH}	50
Table 8 - Shift values for the Robust modes.....	52
Table 9 - Selections from clause 20 of [28] amended by [29]	58
Table 10 - Fields associated with PHY Control Plane Primitives.....	65
Table 11 - Values of the status parameter in PLME_GET.confirm primitive	78
Table 12 - Values of the result parameter in the PLME_SET.confirm primitive.....	79
Table 13 - Broadcast and multicast address.....	86
Table 14 - Direct connection example: Node B's Direct switching table	105
Table 15 - Values of <i>MACUpdateKeysTime</i> for different number of channels	110
Table 16 –Nonce SID/LNID Derivation.....	111
Table 17 – Encryption/Authentication by PDU Types	116
Table 18 - Generic MAC header fields.....	121
Table 19 – Packet header fields.....	123
Table 20 – Security subheader fields.....	125
Table 21 - MAC control packet types	126
Table 22 - REG control packet fields.....	130
Table 23 - REG control packet types.....	136
Table 24 - CON control packet fields	137
Table 25 - CON control packet types.....	139

Table 26 - PRO control packet fields.....	141
Table 27 - PRO control packet types	142
Table 28 - Extension for Multi PHY Promotion.....	144
Table 29 – New fields defined in the MultiPHY PRO messages.....	145
Table 30 - FRA control packet fields	145
Table 31 - CFP control message fields	146
Table 32 - CFP control packet types	147
Table 33 - ALV control message fields.....	148
Table 34 – Keep-Alive control packet types	150
Table 35 - MUL control message fields	151
Table 36 – MUL control message types.....	152
Table 37 – SEC control message fields	154
Table 38 - SEC control packet types	155
Table 39 - PCC control packet fields	155
Table 40 - PCC control message types.....	156
Table 41 - Promotion Need MAC PDU fields	157
Table 42 - Beacon PDU fields.....	159
Table 43 – List of MAC primitives	163
Table 44 – Values of the <i>Answer</i> parameter in MAC_ESTABLISH.response primitive	166
Table 45 – Values of the <i>Result</i> parameter in MAC_ESTABLISH.confirm primitive	167
Table 46 – Values of the <i>Reason</i> parameter in MAC_RELEASE.indication primitive.....	168
Table 47 – Values of the <i>Answer</i> parameter in MAC_RELEASE.response primitive.....	168
Table 48 – Values of the <i>Result</i> parameter in MAC_RELEASE.confirm primitive.....	169
Table 49 – Values of the <i>Result</i> parameter in MAC_JOIN.confirm primitive	170
Table 50 – Values of the <i>Answer</i> parameter in MAC_ESTABLISH.response primitive	172
Table 51 – Values of the <i>Result</i> parameter in MAC_LEAVE.confirm primitive.....	173
Table 52 – Values of the <i>Result</i> parameter in MAC_DATA.confirm primitive.....	175

Table 53 – Values of the <i>Result</i> parameter in MLME_REGISTER.confirm primitive	176
Table 54 – Values of the <i>Result</i> parameter in MLME_UNREGISTER.confirm primitive	178
Table 55 - Values of the BCN_MODE parameter in MLME_PROMOTE.request primitive.....	179
Table 56 – Values of the <i>Result</i> parameter in MLME_PROMOTE.confirm primitive	179
Table 57 – Values of the <i>Result</i> parameter in MLME_DEMOTE.confirm primitive.....	182
Table 58 - Values of the <i>Result</i> parameter in MLME_MP_DEMOTE.confirm primitive	183
Table 59 – Values of the <i>Result</i> parameter in MLME_RESET.confirm primitive	184
Table 60 – Values of the <i>status</i> parameter in MLME_GET.confirm primitive.....	185
Table 61 – Values of the <i>status</i> parameter in MLME_LIST_GET.confirm primitive	186
Table 62 – Values of the <i>Result</i> parameter in MLME_SET.confirm primitive	187
Table 63 - ARQ fields	226
Table 64 - Time Reference subheader fields	228
Table 65 - SAR header fields	233
Table 66 – SAR.CRC.....	233
Table 67 - SAR Constants.....	234
Table 68 - Primitive mapping between the Null SSCS primitives and the MAC layer primitives	235
Table 69 - IPV4 SSCS Table Entry	238
Table 70 - Mapping IPv4 Precedence to PRIME MAC priority.....	240
Table 71 - AR_REGISTER_S message format	240
Table 72 - AR_REGISTER_B message format	241
Table 73 - AR_UNREGISTER_S message format	241
Table 74 - AR_UNREGISTER_B message format.....	241
Table 75 - AR_LOOKUP_S message format	242
Table 76 - AR_LOOKUP_B message format.....	242
Table 77 - AR_MCAST_REG_S message format.....	242
Table 78 - AR_MCAST_REG_B message format	243
Table 79 - AR_MCAST_UNREG_S message format.....	243

Table 80 - AR_MCAST_UNREG_B message format	243
Table 81 - IPv4 Packet format without negotiated header compression	244
Table 82 - IPv4 Packet format with VJ header compression negotiated.....	244
Table 83 - Connection data sent by the initiator	245
Table 84 - Connection data sent by the responder	245
Table 85 - Connection Data sent by the Service Node	252
Table 86 - Connection Data sent by the Base Node	253
Table 87 – Node addresses table entry	260
Table 88– IPv6 convergence layer table entry	260
Table 89 – Ipv6 convergence layer multicast table entry.....	262
Table 90 – Mapping Ipv6 precedence to PRIME MAC priority	263
Table 91 - AR_REGISTERv6_S message format.....	264
Table 92 - AR_REGISTERv6_B message format	264
Table 93 - AR_UNREGISTERv6_S message format	264
Table 94 - AR_UNREGISTERv6_B message format	265
Table 95 - AR_LOOKUPv6_S message format	265
Table 96 - AR_LOOKUPv6_B message format	265
Table 97 - IPv6 Packet format without header compression	266
Table 98 - UDP/IPv6 Packet format with LOWPAN_IPHC1 header compression and LOWPAN_NHC.....	266
Table 99 - IPv6 Packet format with LOWPAN_IPHC negotiated header compression.....	267
Table 100 - IPv6 Unicast connection data sent by the initiator	267
Table 101 - IPv6 Multicast connection data sent by the initiator	268
Table 102 - PHY read-only variables that provide statistical information.....	275
Table 103 - PHY read-only parameters, providing information on specific implementation	277
Table 104 - PHY read-only parameters of RF Layer, providing information on specific implementation.....	278
Table 105 - Configuration Parameters RF SUN FSK phy layer	278
Table 106 - Table of MAC read-write variables	279

Table 107 - Table of MAC read-only variables.....	285
Table 108 - Table of MAC read-only variables that provide functional information	288
Table 109 - Table of MAC read-only variables that provide statistical information	291
Table 110 - Table of read-only lists made available by MAC layer through management interface	292
Table 111 - Read-only lists only available for Multi-PHY extension	298
Table 112 – MAC security attributes	303
Table 113 - Action PIB attributes	303
Table 114 - Applications PIB attributes	313
Table 115 - FU PIB attributes.....	318
Table 116 - FU State Machine	318
Table 117 - Fields of FU_INIT_REQ.....	323
Table 118 - Fields of FU_EXEC_REQ	324
Table 119 - Fields of FU_CONFIRM_REQ.....	325
Table 120 - Fields of FU_STATE_REQ.....	325
Table 121 - Fields of FU_KILL_REQ	326
Table 122 - Fields of FU_STATE_RSP	326
Table 123 – Fields of Exception code	327
Table 124 - Fields of FU_DATA	327
Table 125 - Fields of FU_MISS_REQ.....	328
Table 126 - Fields of FU_MISS_BITMAP	328
Table 127 - Fields of FU_MISS_LIST.....	329
Table 128 - Fields of FU_INFO_REQ.....	330
Table 129 - Infold possible values	331
Table 130 - Fields of FU_INFO_RSP	331
Table 131 - Fields of each entry of InfoData in FU_INFO_RSP	331
Table 132 - Management data frame fields	335
Table 133 - GET PIB Atribubute request fields	336

Table 134 - GET PIB Attribute response fields.....	337
Table 135 – PIB List query format	339
Table 136 – PIB list response format.....	339
Table 137 - PHY PIB common mandatory attributes.....	341
Table 138 - MAC PIB comon mandatory attributes.....	342
Table 139 - Applications PIB common mandotory attributes	342
Table 140 - MAC PIB Base Node mandatory attributes	343
Table 141 - MAC PIB Service Node mandatory attributes.....	343
Table 142 - APP PIB Service Node mandatory attributes.....	344
Table 143 - Examples of CRC-8 calculated for various ASCII strings.....	345
Table 144 - Example of CRC-32.....	345
Table 145 - Example of Concatenation.....	346
Table 146 - Header interleaving matrix.....	349
Table 147 - DBPSK(FEC ON) interleaving matrix.....	349
Table 148 - DQPSK(FEC ON) interleaving matrix.....	349
Table 149 - D8PSK(FEC ON) interleaving matrix.....	350
Table 150 - Table of MAC constants.....	351
Table 151 - TYPE value assignments.....	353
Table 152 - LCID value assignments	353
Table 153 - Result values for Convergence layer primitives	353
Table 154 – Channel 1: List of frequencies used	356
Table 155 – Channel 2: List of frequencies used	357
Table 156 – Channel 3: List of frequencies used	358
Table 157 – Channel 4: List of frequencies used	359
Table 158 – Channel 5: List of frequencies used	360
Table 159 – Channel 6: List of frequencies used	361
Table 160 – Channel 7: List of frequencies used	362

Table 161 – Channel 8: List of frequencies used	363
Table 162 – Compatibility packet header fields	374
Table 163 - CREG control packet fields.....	376
Table 164 - CREG control packet types.....	379
Table 165 - CPRO control packet fields	380
Table 166 - CPRO control packet types	382
Table 167 - CBSI control packet fields	383
Table 168 - CBSI control message types.....	383
Table 169 - CFRA control packet fields.....	384
Table 170 - CFRA control packet types.....	384
Table 171 - CALV control message fields.....	385
Table 172 – Keep-Alive control packet types	385
Table 173 - Beacon PDU fields.....	386
Table 174 - PHY frame types and Payload transmission schemes	396

1 Introduction

This document is the technical specification for the PRIME technology.

1.1 Scope

This document specifies a PHY layer, a MAC layer and a Convergence layer for complexity-effective, narrowband data transmission over electrical power lines that could be part of a Smart Grid system. Additionally, it specifies RF data transmission that extends the addressed Smart Grid scenarios.

1.2 Overview

The purpose of this document is to specify a narrowband data transmission system based on OFDM modulations scheme and a RF system based on SUN FSK (see [28] and [29]) for providing mainly core utility services.

The specification currently describes the following:

- A PHY layer capable of achieving rates of uncoded 1Mbps on the PLC medium (see chapter 3).
- A PHY layer capable of achieving SUN FSK Operating Mode 2 data rates (see [28] and [29]) on the RF medium (see chapter 3)
- A MAC layer for the power line and RF environments (see chapter 4).
- A Convergence layer for adapting several specific services (see chapter 5).
- A Management Plane (see chapter 6)

The specification is written from the transmitter perspective to ensure interoperability between devices and allow different implementations.

1.3 Normative references

The following publications contain provisions which, through reference in this text, constitute provisions of this specification. At the time of publication, the editions indicated were valid. All standards are subject to revision, and parties to agreements based on this Specification are encouraged to investigate the possibility of applying the most recent editions of the following standards:

#	Ref.	Title
[1]	EN 50065-1:2001+A1:2010	Signalling on low-voltage electrical installations in the frequency range 3 kHz to 148,5 kHz - Part 1: general requirements, frequency bands and electromagnetic disturbances.
[2]	EN IEC 50065-7 Ed. 2001	Signalling on low-voltage electrical installations in the frequency range 3 kHz to 148,5 kHz. Part7: Equipment impedance.
[3]	IEC 61334-4-1 Ed.1996	Distribution automation using distribution line carrier systems – Part 4: Data communication protocols – Section 1: Reference model of the communication system.
[4]	IEC 61334-4-32 Ed.1996	Distribution automation using distribution line carrier systems - Part 4: Data communication protocols - Section 32: Data link layer - Logical link control (LLC).
[5]	IEC 61334-4-511 Ed. 2000	Distribution automation using distribution line carrier systems – Part 4-511: Data communication protocols – Systems management – CIASE protocol.
[6]	IEC 61334-4-512, Ed. 1.0:2001	Distribution automation using distribution line carrier systems – Part 4-512: Data communication protocols – System management using profile 61334-5-1 – Management.
[7]	prEN/TS 52056-8-4	Electricity metering data exchange - The DLMS/COSEM suite - Part 8-4: The PLC Orthogonal Frequency Division Multiplexing (OFDM) Type 1 profile.
[8]	IEEE Std 802-2001	IEEE Standard for Local and Metropolitan Area Networks. Overview and Architecture.
[9]	IETF RFC 768	User Datagram Protocol (UDP) [online]. Edited by J. Postel. August 1980. Available from: https://www.ietf.org/rfc/rfc768.txt
[10]	IETF RFC 791	Internet Protocol (IP) [online]. Edited by Information Sciences Institute, University of Southern California. September 1981. Available from: https://www.ietf.org/rfc/rfc791.txt

#	Ref.	Title
[11]	IETF RFC 793	Transmission Control Protocol (TCP) [online]. Edited by Information Sciences Institute, University of Southern California. September 1981. Available from: https://www.ietf.org/rfc/rfc793.txt
[12]	IETF RFC 1144	Compressing TCP/IP Headers for Low-Speed Serial Links [online]. Edited by V. Jacobson. February 1990. Available from: https://www.ietf.org/rfc/rfc1144.txt .
[13]	IETF RFC 2131	Dynamic Host Configuration Protocol (DHCP) [online]. Edited by R. Droms. March 1997. Available from: https://www.ietf.org/rfc/rfc2131.txt
[14]	IETF RFC 2460	Internet Protocol, Version 6 (IPv6) Specification [online]. Edited by S. Deering, R. Hinden. December 1998. Available from: https://www.ietf.org/rfc/rfc2460.txt
[15]	IETF RFC 3022	Traditional IP Network Address Translator (Traditional NAT) [online]. Edited by P. Srisuresh, Jasmine Networks, K. Egevang. January 2001. Available from: https://www.ietf.org/rfc/rfc3022.txt
[16]	NIST FIPS-197	Specification for the ADVANCED ENCRYPTION STANDARD (AES), http://www.csrc.nist.gov/publications/fips/fips197/fips-197.pdf
[17]	NIST SP 800-57	Recommendation for Key Management. Part 1: General (Revised). Available from http://csrc.nist.gov/publications/nistpubs/800-57/sp800-57-Part1-revised2_Mar08-2007.pdf
[18]	NIST SP800-38A, Ed. 2001	Recommendation for Block Cipher Modes of Operation. Methods and Techniques. Available from http://csrc.nist.gov/publications/nistpubs/800-38a/sp800-38a.pdf .
[19]	IETF RFC 4191	IP version 6 addressing architecture. Available from http://tools.ietf.org/html/rfc4291 .
[20]	IETF RFC 6282	IPv6 Datagrams on IEEE 802.15.4. Available from http://tools.ietf.org/html/rfc6282 .
[21]	IETF RFC 4862	Stateless Address Configuration. Available from http://www.ietf.org/rfc/rfc4862.txt .
[22]	IETF RFC 2464	Transmission of IPv6 Packets over Ethernet Networks. Available from http://www.ietf.org/rfc/rfc4862.txt

#	Ref.	Title
[23]	NIST SP 800-108	Recommendation for Key Derivation Using Pseudorandom Functions. Available from http://csrc.nist.gov/publications/nistpubs/800-108/sp800-108.pdf
[24]	NIST SP 800-38 B	Recommendation for Block Cipher Modes of Operation: The CMAC Mode for Authentication. Available from http://csrc.nist.gov/publications/nistpubs/800-38B/SP_800-38B.pdf
[25]	NIST SP 800-38 C	Recommendation for Block Cipher Modes of Operation: the CCM Mode for Authentication and Confidentiality. Available from http://csrc.nist.gov/publications/nistpubs/800-38C/SP800-38C_updated-July20_2007.pdf
[26]	NIST SP 800-38 F	Recommendation for Block Cipher Modes of Operation: Methods for Key Wrapping. Available from http://dx.doi.org/10.6028/NIST.SP.800-38F
[27]	DRAFT NIST SP 800-90 C	Recommendation for Random Bit Generator (RBG) Constructions. Available from: http://csrc.nist.gov/publications/drafts/800-90/draft-sp800-90c.pdf
[28]	802.15.4-2015	IEEE Standard for Low-Rate Wireless Networks. Available from: https://standards.ieee.org/content/ieee-standards/en/standard/802_15_4-2015.html
[29]	802.15.4v-2017	Amendment 5: Enabling/Updating the Use of Regional Sub-GHz Bands. Available from: https://standards.ieee.org/content/ieee-standards/en/standard/802_15_4v-2017.html
[30]	DLMS UA 1000-2 ED.12	DLMS/COSEM Architecture and Protocols. Available from: https://www.dlms.com/produit/dlms-ua-specifications-green-book-ed-12/ DLMS UA 1000-2 ED.12

1.4 Document conventions

This document is divided into chapters and annexes. The document body (all chapters) is normative (except for italics). The annexes may be normative or Informative as indicated for each annex.

Binary numbers are indicated by the prefix '0b' followed by the binary digits, e.g. '0b0101'. Hexadecimal numbers are indicated by the prefix '0x'.

Mandatory requirements are indicated with 'shall' in the main body of this document.

Optional requirements are indicated with 'may' in the main body of this document. If an option is incorporated in an implementation, it shall be applied as specified in this document.

roof (.) denotes rounding to the closest higher or equal integer.

floor (.) denotes rounding to the closest lower or equal integer.

A mod B denotes the remainder (from 0, 1, ..., B-1) obtained when an integer A is divided by an integer B.

1.5 Definitions

Term	Description
Band	For PLC, set of channels that may or may not be adjacent but defined for concurrent use according to channel access rules laid down in this specification. For RF, set of channels that cannot be used concurrently in a unique transmission/reception.
Band Plan	Set of bands that a device is configured to operate on.
Base Node	Master Node which controls and manages the resources of a Subnetwork.
Beacon Slot	Location of the beacon PDU within a frame.
Channel	For PLC, 46.875 kHz spectrum that may either correspond to PRIME version 1.3.6 spectrum location or any of the new extension bands defined in this version of specification. For RF, spectrum that is used in a unique transmission/reception. The spectrum width depends on the RF band the channel belongs to and on the operating mode.
Compliance Mode	A working mode of MAC protocol that supports existence of legacy 1.3.6 devices in a Subnetwork together with devices implementing this version of specification.
Destination Node	A Node that receives a frame.
Downlink	Data travelling in direction from Base Node towards Service Nodes
Hearing Domain	Area in which transmit signal from a device is received with some fidelity, without the need of intermediate amplification/repeating devices.
Level(PHY layer)	When used in physical layer (PHY) context, it implies the transmit power level.
Level (MAC layer)	When used in medium access control (MAC) context, it implies the position of the reference device in Switching hierarchy.

Term	Description
MAC frame	Composite unit of abstraction of time for channel usage. A MAC frame is comprised of one or more Beacons, one SCP and zero or one CFP. The transmission of the Beacon by the Base Node acts as delimiter for the MAC frame.
Neighbour Node	Node A is Neighbour Node of Node B if A can directly transmit to and receive from B.
Node	Any one element of a Subnetwork which is able to transmit to and receive from other Subnetwork elements.
PHY frame	The set of OFDM symbols and Preamble which constitute a single PPDU
Peer	Two devices within the hearing domain of each other and having possibility or maintaining data-connectivity with each other without need of intermediate repeater / switch devices.
Preamble	The initial part of a PHY frame, used for synchronizations purposes
Registration	Process by which a Service Node is accepted as member of Subnetwork and allocated a LNID.
Service Node	Any one Node of a Subnetwork which is not a Base Node.
Source Node	A Node that sends a frame.
Subnetwork	A set of elements that can communicate by complying with this specification and share a single Base Node.
Subnetwork address	Property that universally identifies a Subnetwork. It is its Base Node EUI-48 address.
Switching	Providing connectivity between Nodes that are not Neighbour Nodes.
Unregistration	Process by which a Service Node leaves a Subnetwork.
Uplink	Data travelling in direction from Service Node towards Base Node

39

1.6 Abbreviations and Acronyms

40

Term	Description
AC	Alternating Current
AES	Advanced Encryption Standard
AMM	Advanced Meter Management
ARQ	Automatic Repeat Request

Term	Description
ATM	Asynchronous Transfer Mode
BER	Bit Error Rate
BPDU	Beacon PDU
BPSK	Binary Phase Shift Keying
CENELEC	European Committee for Electrotechnical Standardization
CFP	Contention Free Period
CID	Connection Identifier
CL	Convergence layer
CPCS	Common Part Convergence Sublayer
CRC	Cyclic Redundancy Check
CSMA-CA	Carrier Sense Multiple Access-Collision Avoidance
D8PSK	Differential Eight-Phase Shift Keying
DBPSK	Differential Binary Phase Shift Keying
DHCP	Dynamic Host Configuration Protocol
DPSK	Differential Phase Shift Keying (general)
DQPSK	Differential Quaternary Phase Shift Keying
DSK	Device Secret Key
ECB	Electronic Code Book
EMA	Exponential moving average
ENOB	Effective Number Of Bits
EUI-48	48-bit Extended Unique Identifier
EVM	Error Vector Magnitude
FCS	Frame Check Sequence
FEC	Forward Error Correction
FFT	Fast Fourier Transform

Term	Description
FSK	Frequency Shift Keying
GK	Generation Key
GPDU	Generic MAC PDU
HCS	Header Check Sum
IEC	International Electrotechnical Committee
IEEE	Institute of Electrical and Electronics Engineers
IFFT	Inverse Fast Fourier Transform
IGMP	Internet Group Management Protocol
IPv4	Internet Protocol, Version 4
kbits	kilobit per second
KDIV	Key Diversifier
LCID	Local Connection Identifier
LFSR	Linear Feedback Shift Register
LLC	Logical Link Control
LNID	Local Node Identifier
LSID	Local Switch Identifier
LV	Low Voltage
LWK	Local Working Key
MAC	Medium Access Control
MK	Master Key
MLME	MAC Layer Management Entity
MPDU	MAC Protocol Data Unit
msb	Most significant bit
lsb	Least significant bit
MSPS	Million Samples Per Second

Term	Description
MTU	Maximum Transmission Unit
NAT	Network Address Translation
NID	Node Identifier
NSK	Network Secret Key
OFDM	Orthogonal Frequency Division Multiplexing
PDU	Protocol Data Unit
PHY	Physical Layer
PIB	PLC Information Base
PLC	Powerline Communications
PLME	PHY Layer Management Entity
PNPDU	Promotion Needed PDU
PPDU	PHY Protocol Data Unit
ppm	Parts per million
PSD	Power Spectral Density
PSDU	PHY Service Data Unit
QoS	Quality of Service
SAP	Service Access Point
SAR	Segmentation and Reassembly
SCP	Shared Contention Period
SCRC	Secure CRC
SDU	Service Data Unit
SEC	Security
SID	Switch Identifier
SNA	Subnetwork Address
SNK	Subnetwork Key (corresponds to either REG.SNK or SEC.SNK)

Term	Description
SNR	Signal to Noise Ratio
SP	Security Profile
SSCS	Service Specific Convergence Sublayer
SWK	Subnetwork Working Key
TCP	Transmission Control Protocol
TOS	Type Of Service
UI	Unique Identifier
USK	Unique Secret Key
VJ	Van Jacobson
WK	Working Key

2 General Description

2.1 Introduction

This document is the Specification of a communication solution to provide complexity-effective, narrowband data transmission over electrical power lines that could be part of a Smart Grid system, based on PLC using orthogonal frequency division multiplexing (OFDM) modulation scheme or RF using some of the general PHY requirements defined on clause 10 of IEEE 802.15.4-2015 [28], including as mandatory the SUN FSK PHY, defined on clause 20 of IEEE 802.15.4-2015 [28].

2.2 General description of the architecture

Figure 1 below depicts the communication layers and the scope of this specification. This specification focuses mainly on the data, control and management plane.

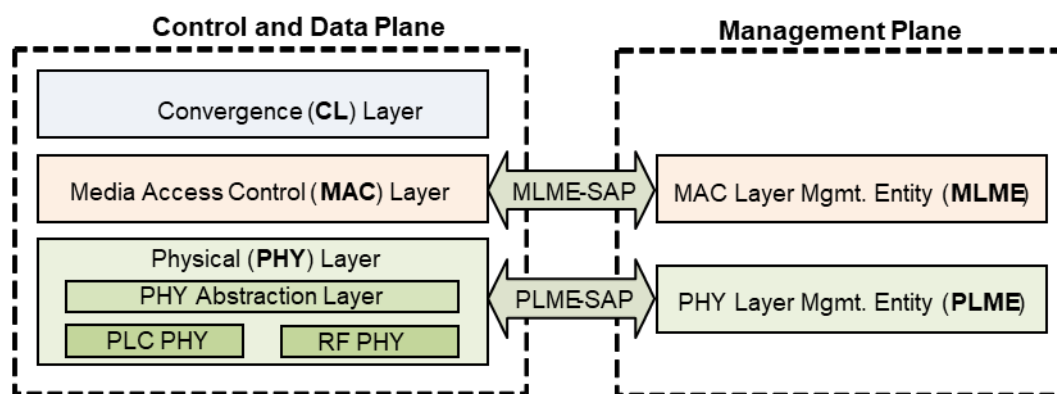


Figure 1 - Reference model of protocol layers used in the PRIME specification

The CL classifies traffic associating it with its proper MAC connection; this layer performs the mapping of any kind of traffic to be properly included in MPDUs. It may also include header compression functions. Several SSCs are defined to accommodate different kinds of traffic into MPDUs.

The MAC layer provides core MAC functionalities of system access, bandwidth allocation, connection establishment/maintenance and topology resolution.

The PHY layer transmits and receives MPDUs between Neighbor Nodes using PLC or RF depending on the capability of devices and characteristics of the medium. On PLC, OFDM is chosen as the modulation technique because of:

- its inherent adaptability in the presence of frequency selective channels (which are common but unpredictable, due to narrowband interference or unintentional jamming);
- its robustness to impulsive noise, resulting from the extended symbol duration and use of FEC;
- its capacity for achieving high spectral efficiencies with simple transceiver implementations.

The PHY specification, described in Chapter 3, also employs a flexible coding scheme. The PHY data rates can be adapted to channel and noise conditions by the MAC.

3 Physical layer

3.1 Introduction

This chapter specifies the PHY entity employed to ensure the transmission and reception of MPDUs between Neighbor Nodes using PLC or RF depending on the capability of devices and characteristics of the medium.

The PHY Abstraction Layer provides a common interface for PLC or RF PHYs. On the transmission of MPDUs the PHY Abstraction Layer uses PLC PHY or RF PHY depending of the selection of the MAC layer. On the reception of a MPDU the PHY Abstraction Layer reports to the MAC layer the MPDU received, the reception condition, the PHY medium, and the band (PLC) or the channel (RF) of the medium where the MPDU was received.

The PLC PHY entity uses frequencies in the band 3 kHz up to 500 kHz. The use of these frequencies is subject to applicable local regulations, e.g. EN 50065 1:2001+A1:2010 in Europe or FCC part 15 in the US.

It is well known that frequencies below 40 kHz show several problems in typical LV power lines. For example:

- load impedance magnitude seen by transmitters is sometimes below 1Ω , especially for Base Nodes located at transformers;
- colored background noise, which is always present in power lines and caused by the summation of numerous noise sources with relatively low power, exponentially increases its amplitude towards lower frequencies;
- meter rooms pose an additional problem, as consumer behaviors are known to have a deeper impact on channel properties at low frequencies, i.e. operation of all kind of household appliances leads to significant and unpredictable time-variance of both the transfer function characteristics and the noise scenario.

Consequently, the PRIME PLC PHY specification uses the frequency band from 41.992 kHz to 471.6796875 kHz. This range is divided into eight channels, which may be used either as single independent channels or “N_{CH}” of them concurrently as a unique transmission / reception band. OFDM modulation is specified in each channel, with signal loaded on 97 equally spaced subcarriers, transmitted in symbols of 2240 microseconds, of which 192 microseconds are comprised of a short cyclic prefix. Adjacent channels are always separated by guard intervals of fifteen subcarriers (7.3 kHz). More details are provided in Annex G.

Differential modulations are used, with one of three possible constellations: DBPSK, DQPSK or D8PSK.

An additive scrambler is used to avoid the occurrence of long sequences of identical bits.

Finally, $\frac{1}{2}$ rate convolutional coding and repetition code will be used along with bit interleaving. The convolutional coding, the bit interleaving and/or the repetition code can be disabled by higher layers if the channel is good enough and higher throughputs are needed.

In addition to the PLC PHY, the PHY layer can use an RF PHY to ensure the communication of the entire network, when the media conditions or the characteristics of the device require it. At each point-to-point connection, the MAC layer can select the PHY layer, and the bands or channels available in it, that are most convenient to ensure the best communication.

3.2 Overview

3.2.1 General

On the transmitter side, the PHY Layer receives a MPDU from the MAC layer and the PHY Abstraction Layer uses the PLC PHY or RF PHY to generates a PHY frame on the Physical medium according to the selection of the MAC layer.

In the case that the PHY Abstraction Layer uses the PLC PHY, the header and the PPDU is shown in Figure 2, and consists of the following steps.

A CRC is appended to the PHY header (CRC for the payload is appended by the MAC layer, so no additional CRC is inserted by the PHY). Next, CC is performed, if the optional FEC is enabled. The next step is scrambling, which is done for both PHY header and the PPDU, irrespective of whether CC is enabled. When CC is enabled, additional repetition code can be selected and, in this case, the scrambler output is repeated by a factor of four. This transmitter configuration is defined as robust mode. If CC is enabled, the scrambler output (or the repeater output in case of robust modes) is also interleaved.

The scrambled (and interleaved) bits are differentially modulated using a DBPSK, DQPSK or D8PSK scheme. The next step is OFDM, which comprises the IFFT block and the cyclic prefix generator. When header and data bits are input to the chain shown in Figure 2, the output of the cyclic prefix generation is a concatenation of OFDM symbols constituting the header and payload portions of the PPDU respectively. The header portion contains two or four OFDM symbols, while the payload portion contains M OFDM symbols. The value of M is signaled in the PHY header, as described in Section 3.3.2.3



Figure 2 - Overview of PPDU processing

Two different PHY frame formats are specified, named frame of Type A and Type B. The structure of the PRIME PHY frame of Type A is shown in Figure 3. Each PHY frame of Type A starts with a preamble lasting 2.048 ms, followed by a number of OFDM symbols, each lasting 2.24 ms. The first two OFDM symbols carry the PHY frame header also referred to as the header in this specification. The header is also generated from using a process similar to the payload generation, as described in Section 3.3.2.3.1. The remaining M OFDM symbols carry payload, generated as described in Section 3.3.2.3.1. The value of M is signaled in the header and is at most equal to 63.

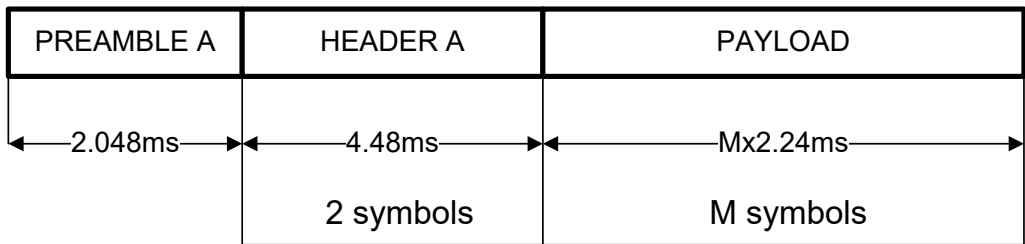


Figure 3 - PHY frame of Type A format

The structure of the PHY frame of Type B is shown in Figure 4. Each PHY frame of Type B starts with a preamble lasting 8.192 ms, followed by a number of OFDM symbols, each lasting 2.24 ms. The first four OFDM symbols carry the PHY frame header. The header is also generated from using a process similar to the payload generation, as described in Section 3.3.2.3.2. The remaining M OFDM symbols carry payload, generated as described in Section 3.3.2.3.2. The value of M is signaled in the header, and is at most equal to 252.

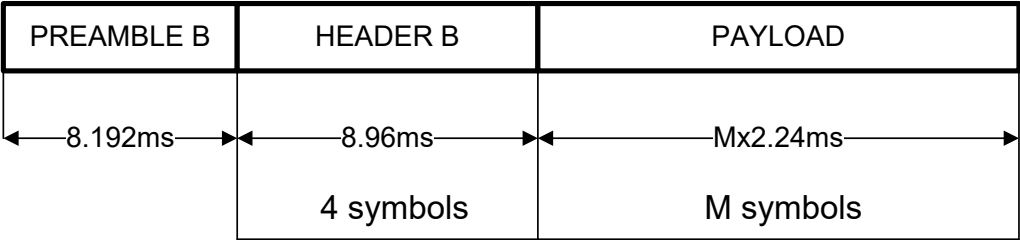


Figure 4 - PHY frame of Type B format

When the PHY Abstraction Layer uses the RF PHY, the PPDU format and processing are respectively described in clause 5.7.3 and 20.2 and 20.3 of IEEE 802.15.4 2015 [28] in the case of SUN FSK. They will be supported with the restrictions indicated throughout this specification (see in particular Section 3.4).

3.2.2 Note about backwards compatibility with PRIME v1.3.6

The current version of the PRIME specification includes new features and several modifications at PHY level. In order to ensure backwards compatibility with deployed PRIME devices, which shall be compliant with previous PRIME specification version 1.3.6, a “PHY backwards compatibility” mechanism is described in Annex J.

3.3 PLC PHY

3.3.1 PHY parameters

Table 1 lists the frequency and timing parameters used in the PRIME PHY. These parameters are common for all constellation/coding combinations.

Note: Note that throughout this document, a sampling rate of 1 MHz and 2048-point FFT sizes are defined for specification convenience of the OFDM signals and are not intended to indicate a requirement on the implementation

Table 1 - Frequency and Timing Parameters of the PRIME PHY

Parameter	Values	
Base Band clock (Hz)	1000000	
Subcarrier spacing (Hz)	488.28125	
Number of data subcarriers	N _{CH} ×84 (header)	N _{CH} ×96 (payload)

Parameter	Values	
Number of pilot subcarriers	$N_{CH} \times 13$ (header)	$N_{CH} \times 1$ (payload)
FFT interval (samples)	2048	
FFT interval (μs)	2048	
Cyclic Prefix (samples)	192	
Cyclic Prefix (μs)	192	
Symbol interval (samples)	2240	
Symbol interval (μs)	2240	
Preamble period (μs)	2048 (Type A)	8192 (Type B)

159

160 **Note:** $1 \leq N_{CH} \leq 8$, where " N_{CH} " is the number of channels as defined in Section 3.1.

161 Table 2 below shows the PHY data rate during payload transmission, and maximum MSDU length for various
 162 modulation and coding combinations. The robust modes, which include CC and repetition coding, only allow
 163 for DBPSK and DQPSK modulations. The effect of using more than one channel, as defined in Section 3.1, is
 164 represented by " N_{CH} " ($1 \leq N_{CH} \leq 8$).

165

166

Table 2 - PHY Payload Parameters

	DBPSK			DQPSK			D8PSK	
Convolutional Code (1/2)	On	On	Off	On	On	Off	On	Off
Repetition code	On	Off	Off	On	Off	Off	Off	Off
Information bits per subcarrier N_{BPSC}	0.5	0.5	1	1	1	2	1.5	3
Information bits per OFDM symbol N_{BPS}	$N_{CH} \times 48$	$N_{CH} \times 48$	$N_{CH} \times 96$	$N_{CH} \times 96$	$N_{CH} \times 96$	$N_{CH} \times 192$	$N_{CH} \times 144$	$N_{CH} \times 288$
Raw data rate (kbps approx)	$N_{CH} \times 5.4$	$N_{CH} \times 21.4$	$N_{CH} \times 42.9$	$N_{CH} \times 10.7$	$N_{CH} \times 42.9$	$N_{CH} \times 85.7$	$N_{CH} \times 64.3$	$N_{CH} \times 128.6$
Maximum number of payload symbols	252	63	63	252	63	63	63	63

	DBPSK			DQPSK			D8PSK	
Maximum MPDU2 length with the maximum number of payload symbols (in bits)	$N_{CH} \times 3016$	$N_{CH} \times 3016$	$N_{CH} \times 6048$	$N_{CH} \times 6040$	$N_{CH} \times 6040$	$N_{CH} \times 12096$	$N_{CH} \times 9064$	$N_{CH} \times 18144$
Maximum MPDU2 length with the maximum number of payload symbols (in bytes)	$N_{CH} \times 377$	$N_{CH} \times 377$	$N_{CH} \times 756$	$N_{CH} \times 755$	$N_{CH} \times 755$	$N_{CH} \times 1512$	$N_{CH} \times 1133$	$N_{CH} \times 2268$

Table 3 shows the modulation and coding scheme and the size of the header portion of the PHY frame (see Section 3.3.2.3).

Note: The whole MPDU includes MPDU1 and MPDU2. The length of MPDU1 is defined in Section 3.4.3.1 for the Type A frames and Section 3.4.3.2 for Type B frames.

Table 3 – PHY Header Parameters

	DBPSK with Header Type A	DBPSK with Header Type B
Convolutional Code (1/2)	On	On
Repetition code	Off	On
Information bits per subcarrier N_{BPSC}	0.5	0.5
Information bits per OFDM symbol N_{BPS}	$N_{CH} \times 42$	$N_{CH} \times 42$

It is strongly recommended that all frequencies used to generate the OFDM transmit signal come from one single frequency reference. The system clock shall have a maximum tolerance of ± 50 ppm, including ageing.

3.3.2 Preamble, header and payload structure

3.3.2.1 Preamble

3.3.2.1.1 PRIME preamble Type A

The preamble is used at the beginning of every PPDU for synchronization purposes. In order to provide a maximum of energy, a constant envelope signal is used instead of OFDM symbols. There is also a need for the preamble to have frequency agility that will allow synchronization in the presence of frequency selective attenuation and, of course, excellent aperiodic autocorrelation properties are mandatory. A linear chirp sequence meets all the above requirements.

The preamble of Type A, named $S(t)$, is composed by N_{CH} sub-symbols where N_{CH} is the number of channels concurrently used. The set of the active channels indices is defined Ω and its i^{th} element is ω_i .

186 $\Omega = \{\omega \in [1,2,...,8]: \omega \text{ is an active channel}\} = \{\omega_1, \omega_2, ..., \omega_{N_{CH}}\}$

187 The preamble sub-symbol $S_{SS}^c(t)$ contains a chirp signal ranging on the frequencies of channel c as defined
188 in Annex G:

189
$$S_{SS}^c(t) = B \cdot \text{window}(t/T') \cdot \cos[2\pi(f_0^c t + 1/2 \mu_c t^2)] \quad 0 \leq t < T'$$

190 where T' is the duration of the chirp, $\mu_c = (f_f^c - f_0^c)/T'$, f_0^c and f_f^c are the start and final frequencies of
191 the channel c , respectively. The function $\text{window}(t/T')$ is a shaping window of length T' composed by a raising
192 roll-off region with length ro μ s a flat region (of unitary amplitude) and a decreasing roll-off region with length
193 ro μ s. The definition of the roll-off region shape is left to individual implementations and should aim at
194 reducing the out-of-band spectral emissions.

195 The choice of the parameter B determines the average preamble power that must be 4 dB higher than the
196 average power of the header and payload OFDM symbols.

197 The duration T' of the sub-symbols, in μ s, is defined as follows:

198
$$T' = \frac{2048 - ro}{N_{CH}} + ro$$

199 The preamble $S(t)$ is the concatenation of the sub-symbols $S_{SS}^c(t)$ with their head and tail roll-off regions
200 overlapped:

201
$$S(t) = \sum_{i=0}^{N_{CH}-1} S_{SS}^{\omega_i}(t - i \cdot (T' - ro)) \quad 0 \leq t < (T' - ro) \cdot N_{CH} + ro$$

202 To avoid rounding issues in the definition of T' , the length of the roll-off region depends on the number of
203 active channels N_{CH} and its values are listed in Table 4.

204 **Table 4 - Roll-off region length for all N_{CH} values**

N_{CH}	ro [μ s]	N_{CH}	ro [μ s]
1	0	5	63
2	64	6	62
3	62	7	67
4	64	8	64

205

206 Note that when a single channel is used the roll-off regions are not present, $T' = 2048$ μ s and $S(t) \equiv S_{SS}^c(t)$
207 .

208 Figure 5 is an example of the structure of the preamble $S(t)$ when three channels are used (channel 1,
209 channel 3 and channel 6). In this case, $N_{CH} = 3$, $ro = 62$ μ s and $T' = 724$ μ s.

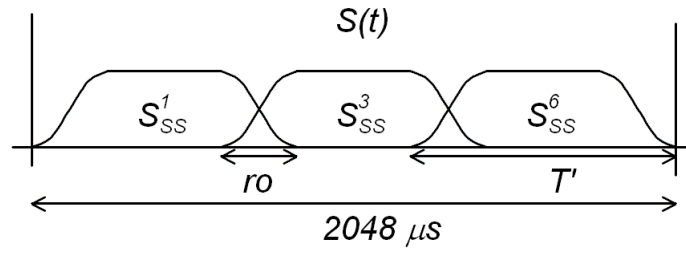


Figure 5 - Example of the preamble structure when three channels are used

3.3.2.1.2 PRIME preamble Type B

The preamble of Type B, named $S(t)$, is the concatenation of three preamble symbols $S_{PS}(t)$ and one preamble symbol with inverted sign $-S_{PS}(t)$ as shown in Figure 6.

$$S(t) = \begin{bmatrix} S_{PS}(t) & S_{PS}(t) & S_{PS}(t) & -S_{PS}(t) \end{bmatrix}$$

Figure 6 - Preamble Type B structure

Each preamble symbol $S_{PS}(t)$ is composed by N_{CH} sub-symbols $S_{SS}(t)$ where N_{CH} is the number of channels concurrently used. The set of the active channels indices is defined Ω and its i^{th} element is ω_i .

$$\Omega = \{\omega \in [1, 2, \dots, 8]: \omega \text{ is an active channel}\} = \{\omega_1, \omega_2, \dots, \omega_{N_{CH}}\}$$

The sub-symbol $S_{SS}^c(t)$ contains a chirp signal ranging on the frequencies of channel c as defined in Annex G:

$$S_{SS}^c(t) = B \cdot \text{window}(t/T') \cdot \cos[2\pi(f_0^c t + 1/2 \mu_c t^2)] \quad 0 \leq t < T'$$

where T' is the duration of the chirp, $\mu_c = (f_f^c - f_0^c)/T'$, f_0^c and f_f^c are the final and start frequencies of the channel c , respectively. The function $\text{window}(t/T')$ is a shaping window of length T' composed by a raising roll-off region with length $ro \mu s$, a flat region (of unitary amplitude) and a decreasing roll-off region with length $ro \mu s$. The definition of the roll-off region shape is left to individual implementations and should aim at reducing the out-of-band spectral emissions.

The choice of the parameter B determines the average preamble power that must be 4 dB higher than the average power of the header and payload OFDM symbols.

The duration T' of the sub-symbols, in μs , is defined as follows:

$$T' = \frac{2048 - ro}{N_{CH}} + ro$$

The preamble symbol $S_{PS}(t)$ is the concatenation of the sub-symbols $S_{SS}^c(t)$ with their head and tail roll-off regions overlapped:

$$S_{PS}(t) = \sum_{i=0}^{N_{CH}-1} S_{SS}^{\omega_i}(t - i \cdot (T' - ro)) \quad 0 \leq t < (T' - ro) \cdot N_{CH} + ro$$

To avoid rounding issues in the definition of T' , the length of the roll-off region depends on the number of active channels N_{CH} and its values are listed in Table 5.

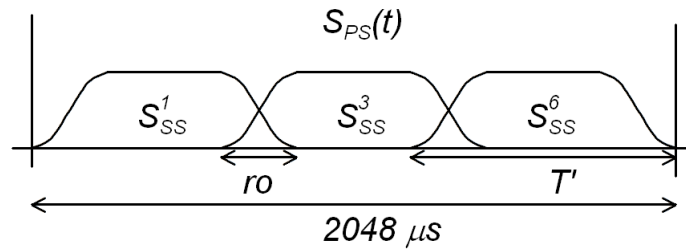
Table 5 - Roll-off region length for all N_{CH} values

N_{CH}	ro [μs]	N_{CH}	ro [μs]
1	0	5	63
2	64	6	62
3	62	7	67
4	64	8	64

237

Note that when a single channel is used the roll-off regions are not present, $T' = 2048 \mu s$ and $S_{PS}(t) \equiv S_{SS}^c(t)$.

Figure 7 is an example of the structure of the preamble symbol $S_{PS}(t)$ when three channels are used (channel 1, channel 3 and channel 6). In this case, $N_{CH} = 3$, $ro = 62 \mu s$ and $T' = 724 \mu s$.



242

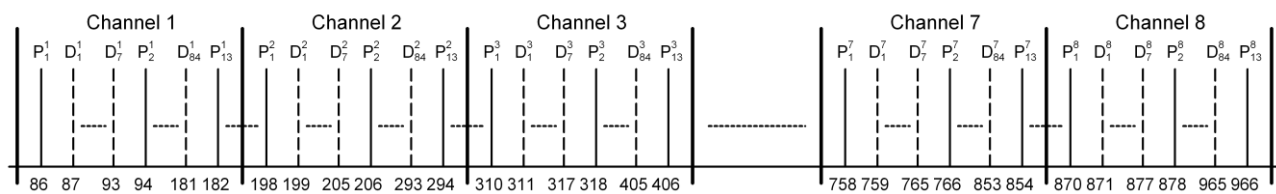
Figure 7 - Example of each of the preamble symbol structure when three channels are used

3.3.2.2 Pilot structure

The preamble is always followed by some OFDM symbols comprising the header. Each header symbol contains $13 \times N_{CH}$ pilot subcarriers, starting from the first subcarrier of each active channel and separated by 7 data subcarriers. The pilots could be used to estimate the sampling start error and the sampling frequency offset.

For subsequent OFDM symbols, one pilot subcarrier is used on the first subcarrier of each active channel to provide a phase reference for frequency domain DPSK demodulation.

In Figure 8 pilot subcarrier allocation is shown for the eight active channels case where a 2048-point FFT is used. P_i^c is the i^{th} pilot subcarrier on the c^{th} channel and D_i^c is the i^{th} data subcarrier on the c^{th} channel.



Subcarrier (2048-point FFT)

Figure 8 - Pilot and data subcarrier frequency allocation inside the header (eight active channels case)

Pilot subcarriers are BPSK modulated by a pseudo-random binary sequence (the pseudo-randomness avoids generation of spectral lines). The phase of the pilot subcarriers is controlled by the sequence p_n , which is a cyclic extension of the 127-bit sequence given by:

$\text{Pref}_{0.126} = \{0,0,0,0,1,1,1,0,1,1,1,0,0,1,0,1,1,0,0,1,0,0,0,0,0,0,1,0,0,0,1,0,0,1,1,0,0,0,1,0,1,1,1,0,1,0,1,1,0,1,1,0,0,0,0,1,1,0,0,1,1,0,1,0,1,0,0,1,1,1,0,0,1,1,1,0,1,1,0,1,0,0,0,1,0,1,0,1,0,1,1,1,1,0,1,0,0,1,0,1,0,0,0,1,0,1,0,1,1,1,1,1,0,1,0,0,1,0,1,0,0,0,1,0,1,0,1,1,1,0,0,0,1,1,1,1,1,1\}$

In the above, '1' means 180° phase shift and '0' means 0° phase shift. One bit of the sequence will be used for each pilot subcarrier, starting with the first pilot subcarrier in the first OFDM symbol, then the next pilot subcarrier, and so on. The same process is used for all the header OFDM symbols. For subsequent OFDM symbols, one element of the sequence is used for the pilot subcarrier of each active channel.

The sequence p_n is generated by the scrambler defined in Figure 9 when the “all ones” initial state is used.

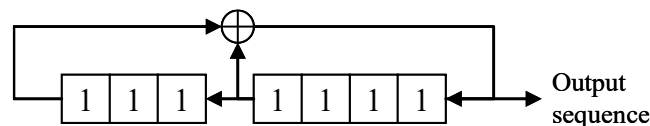


Figure 9 - LFSR for use in Pilot sequence generation

Loading of the sequence pn shall be initiated at the start of every PPDU, just after the Preamble.

3.3.2.2.1 Pilot structure for PHY frames of Type A

In the case of PHY frame of Type A, the header is composed by two OFDM symbols. The pilot and data subcarriers allocation for the eight active channels case is shown in Figure 10.

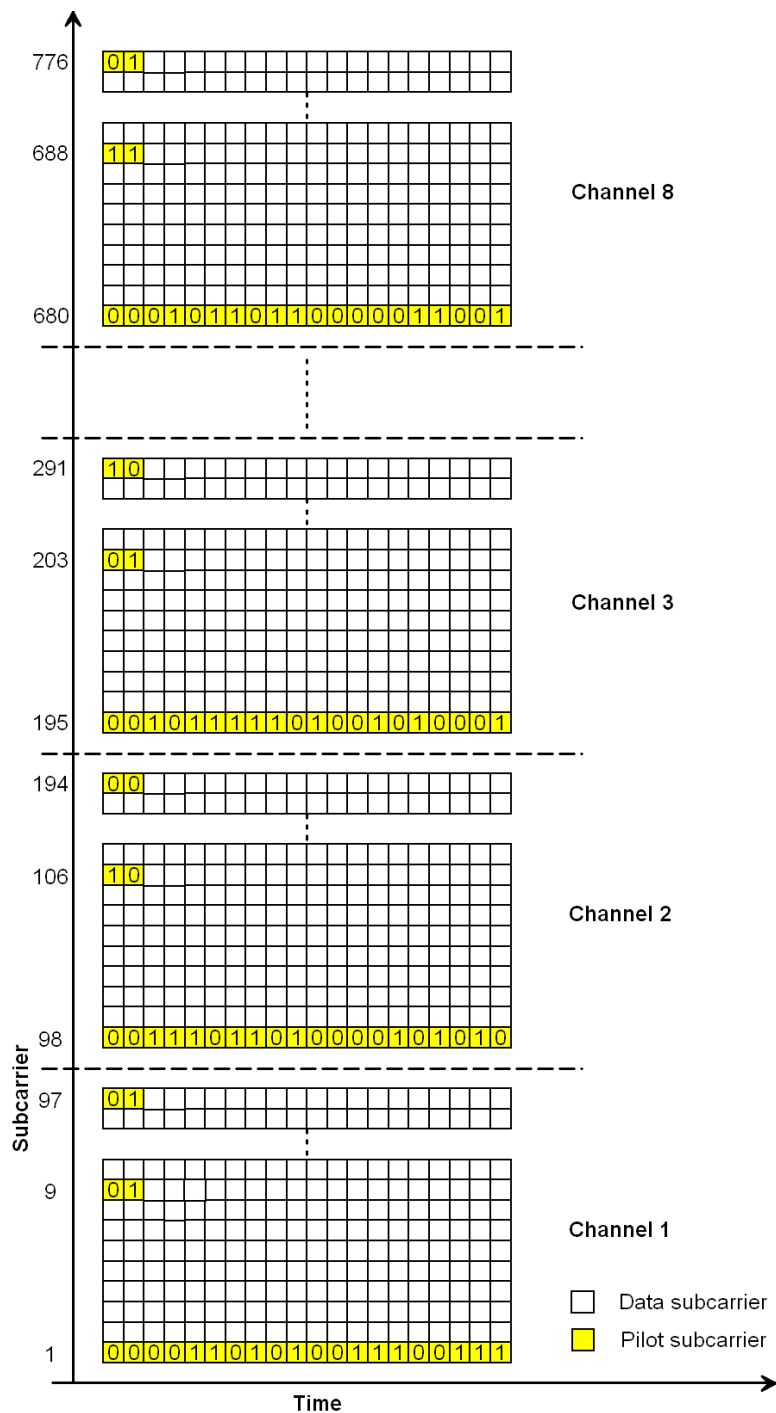


Figure 10 - PHY frame of Type A, pilot and data subcarrier allocation (eight active channels case)

3.3.2.2.2 Pilot structure for PHY frames of Type B

In the case of PHY frame of Type B, the header is composed by four OFDM symbols. The pilot and the data subcarriers allocation for the eight active channels case is shown in Figure 11.

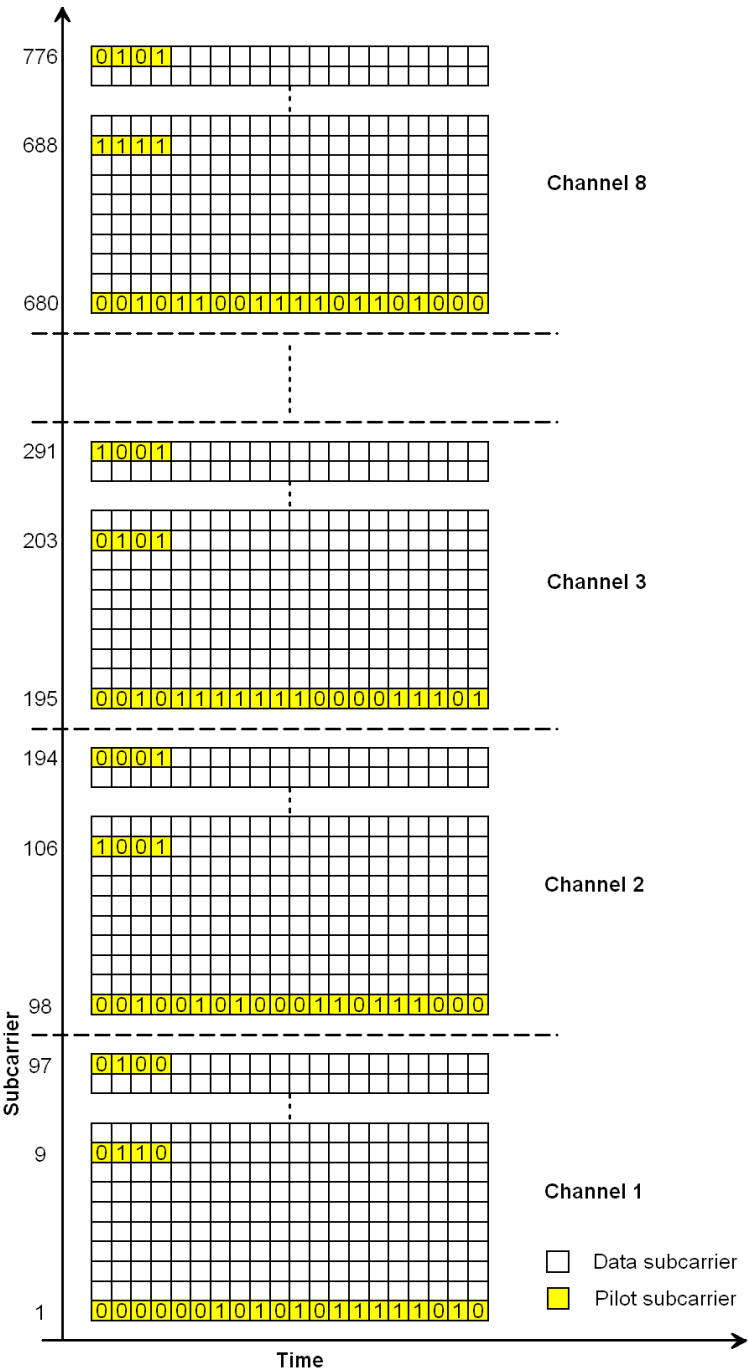


Figure 11 - PHY frame of Type B, pilot and data subcarrier allocation (eight active channels case)

3.3.2.3 Header and Payload

3.3.2.3.1 Header and payload for PHY frames of Type A

The header of Type A is composed of two OFDM symbols, which are always sent using DBPSK modulation and CC "On" (note that the repetition coding is not available for PRIME v1.3.6 devices). The payload is DBPSK, DQPSK or D8PSK modulated, depending on the configuration chosen by the MAC layer. The MAC layer may select the best modulation scheme using information from errors in previous transmissions to the same receiver(s), or by using the SNR feedback. Thus, the system will then configure itself dynamically to provide

the best compromise between throughput and efficiency in the communication. This includes deciding whether or not CC is used.

Note: The optimization metric and the target error rate for the selection of modulation and FEC scheme is left to individual implementations

The first two OFDM symbols in the PPDU (corresponding to the header) are composed of $84 \times N_{CH}$ data subcarriers and $13 \times N_{CH}$ pilot subcarriers. After the header, each OFDM symbol in the payload carries $96 \times N_{CH}$ data subcarriers and one pilot subcarrier. Each data subcarrier carries 1, 2 or 3 bits.

The bit stream from each field must be sent msb first.

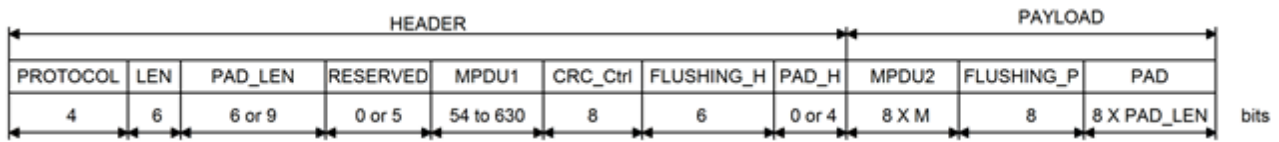


Figure 12 - PRIME PPDU of Type A: header and payload (bits transmitted before encoding)

- HEADER: The header for PRIME PPDUs of Type A comprises two OFDM symbols, containing both PHY and MAC header information. To avoid ambiguity, the MAC header is always referred to as such. The PHY header may also be referred to as just “header”. It is composed of the following fields:

- o PROTOCOL: contains the transmission scheme of the payload. Added by the PHY layer.

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
DBPSK	DQPSK	D8PSK	RES	DBPSK_C	DQPSK_C	D8PSK_C	RES	RES	RES	RES	RES	RS	RES	RES	RES

Where RES means “Reserved” and the suffix “_C” means CC is “On”.

- o LEN: defines the length of the payload (after coding) in OFDM symbols. Added by the PHY layer. If LEN is equal to 0, then the PAYLOAD symbols are not present. In this case, PAD_LEN refers to the padding bytes appended to MPDU bytes to fill the MPDU1 field.
- o PAD_LEN: defines the length of the PAD field (before coding) in bytes. The length in bits of this field depends on the number of active channels: 6 for $N_{CH} = 1$ and 9 for $N_{CH} \geq 2$. Added by the PHY layer.
- o RESERVED: contains the reserved bits for future use. The length in bits of this field depends on the number of active channels: 0 for $N_{CH} = 1$ and 5 for $N_{CH} \geq 2$.
- o MPDU1: First part of the MPDU. The length in bits of this field (MPDU1Len) depends on the number of active channels:

$$MPDU1Len = \left\lfloor \frac{(N_{CH} \cdot 84 - 30) + 2}{8} \right\rfloor \cdot 8 - 2$$

The result of the above formula is reduced by 8 (1 Byte) for $N_{CH} \geq 2$.

- o where $\lfloor x \rfloor$ denotes the nearest integer towards minus infinity of x .
- o CRC_Ctrl: the $CRC\_Ctrl(m)$, $m = 0..7$, contains the CRC checksum over PROTOCOL, LEN, PAD_LEN, RESERVED and MPDU1 field (PD_Ctrl). The polynomial form of PD_Ctrl is expressed as follows:

$$\sum_{m=0}^{69} PD\_Ctrl(m) x^m$$

The checksum is calculated as follows: the remainder of the division of PD_Ctrl by the polynomial $x^8 + x^2 + x + 1$ forms $CRC\_Ctrl(m)$, where $CRC\_Ctrl(0)$ is the lsb. The generator polynomial is the well-known CRC-8-ATM. Some examples are shown in Annex A. Added by the PHY layer.

- FLUSHING_H: flushing bits needed for convolutional decoding. All bits in this field are set to zero to reset the convolutional encoder. Added by the PHY layer.
 - PAD_H: Padding field. In order to ensure that the number of (coded) bits generated in the header fills an integer number of OFDM symbols, pad bits may be added to the header before encoding. All pad bits shall be set to zero. The length in bits of the PAD_H field depends on the number of active channels.
- Table 6 resumes the length in bits of MPDU1 and PAD_H fields for different numbers of active channels.

Table 6 - Length in bits of MPDU1 and PAD_H fields in the PHY frame header of Type A for all possible values of N_{CH}

N_{CH}	MPDU1	PAD_H
1	54	0
2	126	4
3	214	0
4	294	4
5	382	0
6	462	4
7	550	0
8	630	4

- PAYLOAD:

- MPDU2: Second part of the MPDU.
- FLUSHING_P: flushing bits needed for convolutional decoding. All bits in this field are set to zero to reset the convolutional encoder. This field only exists when CC is "On".
- PAD: Padding field. In order to ensure that the number of (coded) bits generated in the payload fills an integer number of OFDM symbols, pad bits may be added to the payload before encoding. All pad bits shall be set to zero.

The MPDU is included in the MPDU1 and MPDU2 fields using the following logic. The first 2 bits of the MPDU are discarded for alignment purposes. The next 54 bits of the MPDU are included in the MPDU1 field. The remaining bits of the MPDU are included in the MPDU2 field. It is a work of higher layers not to use the first two bits of the MPDU as they will not be transmitted or received by the PHY layer. In reception these first non-transmitted bits will be considered as 0.

3.3.2.3.2 Header and payload for PHY frames of Type B

The header is composed of four OFDM symbols, which are always sent using DBPSK modulation, CC "On" and repetition coding "On". However the payload is DBPSK, DQPSK or D8PSK modulated, depending on the configuration by the MAC layer. The MAC layer may select the best modulation scheme using information from errors in previous transmissions to the same receiver(s), or by using the SNR feedback. Thus, the system will then configure itself dynamically to provide the best compromise between throughput and efficiency in the communication. This includes deciding whether or not CC and repetition coding are used.

Note: The optimization metric and the target error rate for the selection of modulation and FEC scheme is left to individual implementations

The first four OFDM symbols in the PPDU (corresponding to the header) are composed of $84 \times N_{CH}$ data subcarriers and $13 \times N_{CH}$ pilot subcarriers. After the header, each OFDM symbol in the payload carries $96 \times N_{CH}$ data subcarriers and N_{CH} pilot subcarriers. Each data subcarrier carries 1, 2 or 3 bits. The bit stream from each field must be sent msb first.

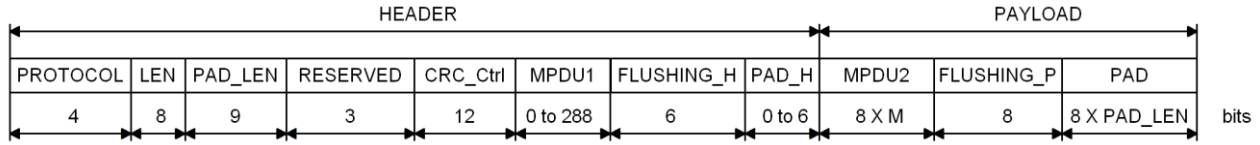


Figure 13 - PRIME PPDU of Type B: header and payload (bits transmitted before encoding)

- HEADER: The PHY header is composed of the following fields:

- PROTOCOL: contains the transmission scheme of the payload. Added by the PHY layer.

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
DBPSK	DQPSK	D8PSK	RES	DBPSK_C	DQPSK_C	D8PSK_C	RES	RES	RES	RES	RES	R_DBPSK	R_DQPSK	RES	RES

Where RES means “Reserved”, the suffix “_C” means CC is “On” and the prefix R_ means CC and repetition are “On”.

- LEN: defines the length of the payload (after coding) in OFDM symbols. Added by the PHY layer.
- PAD_LEN: defines the length of the PAD field (before coding) in bytes. If LEN is equal to 0 the PAYLOAD symbols are not present, in this case PAD_LEN refers to the padding bytes appended to MPDU bytes to fill the MPDU1 field. Added by the PHY layer.
- RESERVED: contains the reserved bits for future use.
- CRC_Ctrl: the $CRC\_Ctrl(m)$, $m = 0..11$, contains the CRC checksum over PROTOCOL, LEN, PAD_LEN and RESERVED field (PD_Ctrl). The polynomial form of PD_Ctrl is expressed as follows:

$$\sum_{m=0}^{23} PD_{Ctrl}(m)x^m$$

The checksum is calculated as follows: the remainder of the division of PD_Ctrl by the polynomial $x^{12} + x^{11} + x^3 + x^2 + x + 1$ forms $CRC\_Ctrl(m)$, where $CRC\_Ctrl(0)$ is the lsb. Some examples are shown in Annex A. Added by the PHY layer.

- MPDU1: First part of the MPDU. The length in bits of this field (MPDU1Len) is a multiple of 8 and it depends on the number of active channels:

$$MPDU1Len = \left\lfloor \frac{(N_{CH} - 1) \cdot 84 \cdot \frac{1}{2}}{8} \right\rfloor \cdot 8$$

where $\lfloor x \rfloor$ denotes the nearest integer towards minus infinity of x.

- FLUSHING_H: flushing bits needed for convolutional decoding. All bits in this field are set to zero to reset the convolutional encoder. Added by the PHY layer.
- PAD_H: Padding field. In order to ensure that the number of (coded) bits generated in the header fills an integer number of OFDM symbols, pad bits may be added to the header before encoding. All pad bits shall be set to zero. The length in bits of PAD_H field (PAD_HLen) depends on the number of active channels:

$$PAD\_HLen = (N_{CH} - 1) \cdot 84 \cdot \frac{1}{2} - MPDU1Len$$

Table 7 resumes the length of MPDU1 and PAD_H fields for different numbers of active channels.

Table 7 - Length in bits of MPDU1 and PAD_H fields in the PHY frame header of Type B for all possible values of N_{CH}

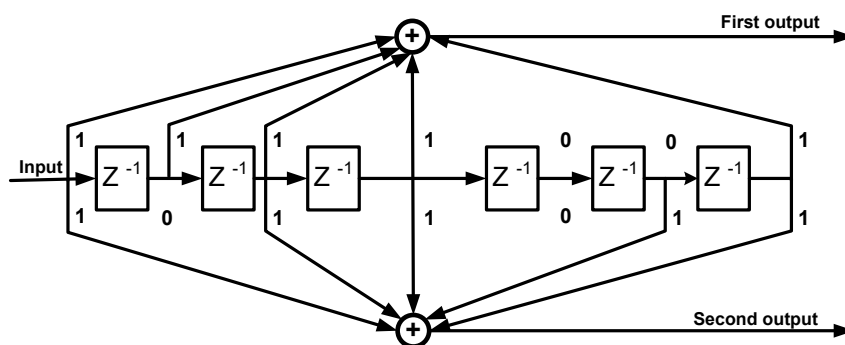
N_{CH}	MPDU1	PAD_H
1	0	0
2	40	2
3	80	4
4	120	6
5	168	0
6	208	2
7	248	4
8	288	6

Note that on reception of a PPDU with a correct CRC_Ctrl but with PROTOCOL with reserved values, or any of the reserved bits being "1", the receiver should consider that the payload contains LEN symbols, and should be able to discard the PDU considering the channel busy.

- PAYLOAD:
 - MPDU2: Second part of the MPDU.
 - FLUSHING_P: flushing bits needed for convolutional decoding. All bits in this field are set to zero to reset the convolutional encoder. This field only exists when CC is "On".
 - PAD: Padding field. In order to ensure that the number of (coded) bits generated in the payload fills an integer number of OFDM symbols, pad bits may be added to the payload before encoding. All pad bits shall be set to zero.

3.3.3 Convolutional encoder

The uncoded bit stream may go through convolutional coding to form the coded bit stream. The convolutional encoder is $\frac{1}{2}$ rate with constraint length $K = 7$ and code generator polynomials 1111001 and 1011011. At the start of every PPDU transmission, the encoder state is set to zero. As seen in Figure 12 and Figure 13, six zeros are inserted at the end of the header information bits to flush the encoder and return the state to zero. Similarly, if convolutional encoding is used for the payload, eight zeros bits are again inserted at the end of the input bit stream to ensure the encoder state returns to zero at the end of the payload. The block diagram of the encoder is shown in Figure 14.


Figure 14 - Convolutional encoder

3.3.4 Scrambler

The scrambler block randomizes the bit stream, so it reduces the crest factor at the output of the IFFT when a long stream of zeros or ones occurs in the header or payload bits after coding (if any). Scrambling is always performed regardless of the modulation and coding configuration.

The scrambler block performs a xor of the input bit stream by a pseudo noise sequence p_n , obtained by cyclic extension of the 127-element sequence given by:

$Pref_{0..126} = \{0,0,0,0,1,1,1,0,1,1,1,1,0,0,1,0,1,1,0,0,1,0,0,1,0,0,0,0,0,1,0,0,0,1,0,0,1,1,0,0,0,1,0,1,1,1,0,1,0,1,1,0,1,1,0,0,0,0,0,1,1,0,0,1,1,0,1,0,1,0,0,0,0,0,1,1,0,0,1,1,0,1,0,0,0,1,1,0,0,0,1,1,0,0,1,1,0,0,0,1,1,1,0,0,1,1,1,0,1,0,0,1,0,0,0,1,0,1,0,1,0,1,1,1,1,0,1,0,0,1,0,1,0,0,0,1,1,0,1,1,0,1,1,1,1,0,1,0,0,1,0,1,0,0,0,1,1,1,1,1,1,1,1\}$

Note: The above 127-bit sequence can be generated by the LFSR defined in Figure 15 when the “all ones” initial state is used.

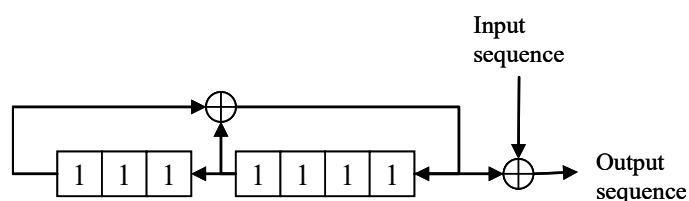


Figure 15 - LFSR for use in the scrambler block

Loading of the sequence p_n shall be initiated at the start of every PPDU, just after the Preamble.

3.3.5 Repeater

The repeater block introduces both time diversity and frequency diversity to the transmitted bits repeating a bit sequence four times with the aim of increasing the communication robustness. The repeater is enabled only when robust modes are used. Figure 16 shows the behavior of the repeater.

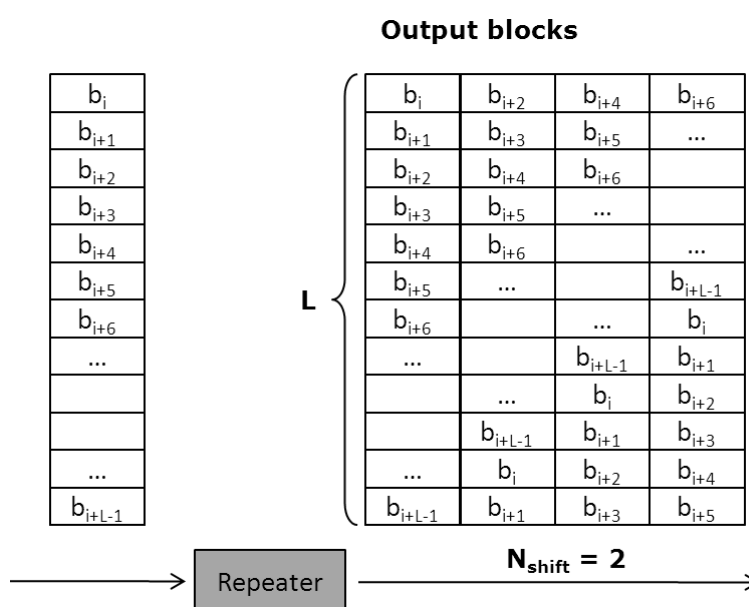


Figure 16 - Example of repeater block using a shift value = 2

The transmitted bit sequence $b_i, b_{i+1}, \dots$ is divided into blocks of length L corresponding to the number of bits transmitted into one OFDM symbol according to the used transmission mode. L is equal to $84 \times N_{CH}$ for the header, $96 \times N_{CH}$ for the payload using robust DBPSK and $192 \times N_{CH}$ for the payload using robust DQPSK, where N_{CH} is the number of channels concurrently used. Each block of L bits is repeated four times at the repeater output. Furthermore, the bits of each replicated block are obtained introducing a cyclic shift of N_{shift} to the bits of the previous block (the first output block always corresponds to the input block). N_{shift} depends on the transmission mode and its values are listed in Table 8.

Table 8 - Shift values for the Robust modes

Transmission mode	N_{shift}
Robust DBPSK (header)	2
Robust DBPSK (payload)	2
Robust DQPSK (payload)	4

3.3.6 Interleaver

Because of the frequency fading (narrowband interference) of typical power line channels, OFDM subcarriers are generally received at different amplitudes. Deep fades in the spectrum may cause groups of subcarriers to be less reliable than others, thereby causing bit errors to occur in bursts rather than be randomly scattered. If (and only if) coding is used as described in 3.4.3, interleaving is applied to randomize the occurrence of bit errors prior to decoding. At the transmitter, the coded bits are permuted in a certain way, which makes sure that adjacent bits are separated by several bits after interleaving.

Let $N_{CBPS} = 2 \times N_{BPS}$ be the number of coded bits per OFDM symbol in the cases convolutional coding is used. All coded bits must be interleaved by a block interleaver with a block size corresponding to N_{CBPS} . The interleaver ensures that adjacent coded bits are mapped onto non-adjacent data subcarriers. Let $v(k)$, with $k = 0, 1, \dots, N_{CBPS} - 1$, be the coded bits vector at the interleaver input. $v(k)$ is transformed into an interleaved vector $w(i)$, with $i = 0, 1, \dots, N_{CBPS} - 1$, by the block interleaver as follows:

$$w((N_{CBPS}/s) \times (k \bmod s) + \text{floor}(k/s)) = v(k) \quad k = 0, 1, \dots, N_{CBPS} - 1$$

The value of s is determined by the number of coded bits per subcarrier, $N_{CBPSC} = 2 \times N_{BPS}$. N_{CBPSC} is related to N_{CBPS} such that $N_{CBPS} = 96 \times N_{CBPSC} \times N_{CH}$ (payload) and $N_{CBPS} = 84 \times N_{CBPSC} \times N_{CH}$ (header), where N_{CH} is the number of channels concurrently used.

$$s = 8 \times (1 + \text{floor}(N_{CBPSC}/2)) \text{ for the payload and}$$

$$s = 7 \text{ for the header.}$$

At the receiver, the de-interleaver performs the inverse operation. Hence, if $w'(i)$, with $i = 0, 1, \dots, N_{CBPS} - 1$, is the de-interleaver vector input, the vector $w'(i)$ is transformed into a de-interleaved vector $v'(k)$, with $k = 0, 1, \dots, N_{CBPS} - 1$, by the block de-interleaver as follows:

$$v'(s \times i - (N_{CBPS} - 1) \times \text{floor}(s \times i / N_{CBPS})) = w'(i) \quad i = 0, 1, \dots, N_{CBPS} - 1$$

Descriptive tables showing index permutations can be found in Annex C for reference.

Note that the interleaver parameters k and N_{CBPS} do not depend on the presence of the repetition encoding and their values remain the same for coded DBPSK (or coded DQPSK) and robust DBPSK (or robust DQPSK).

3.3.7 Modulation

The PPDU payload is modulated as a multicarrier differential phase shift keying signal with one pilot subcarrier and $96 \times N_{CH}$ data subcarriers that comprise $96 \times N_{CH}$, $192 \times N_{CH}$ or $288 \times N_{CH}$ bits per symbol. The header is modulated DBPSK with $13 \times N_{CH}$ pilot subcarriers and $84 \times N_{CH}$ data subcarriers that comprise $84 \times N_{CH}$ bits per symbol.

The bit stream coming from the interleaver is divided into groups of B bits where the first bit of the group of B is the most significant bit (msb).

First of all, frequency domain differential modulation is performed. Figure 17 shows the DBPSK, DQPSK and D8PSK mapping:

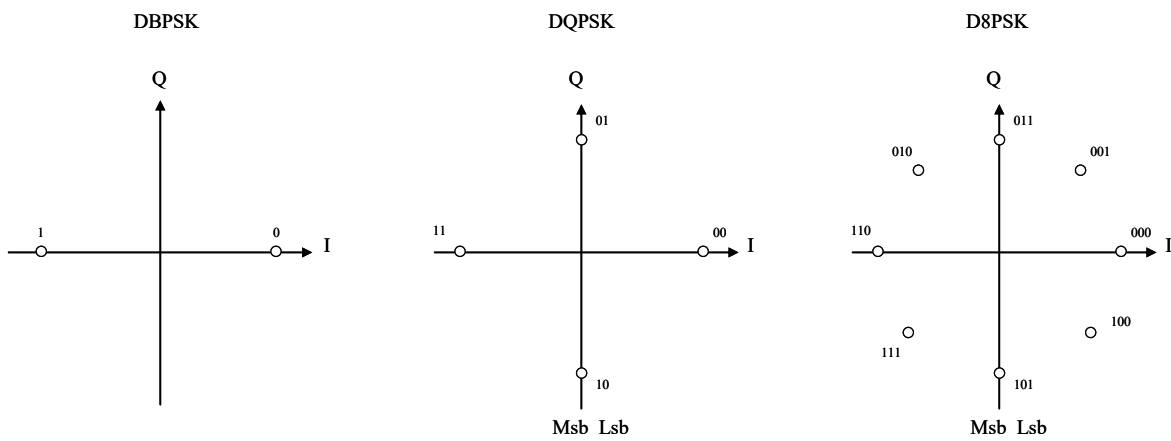


Figure 17 - DBPSK, DQPSK and D8PSK mapping

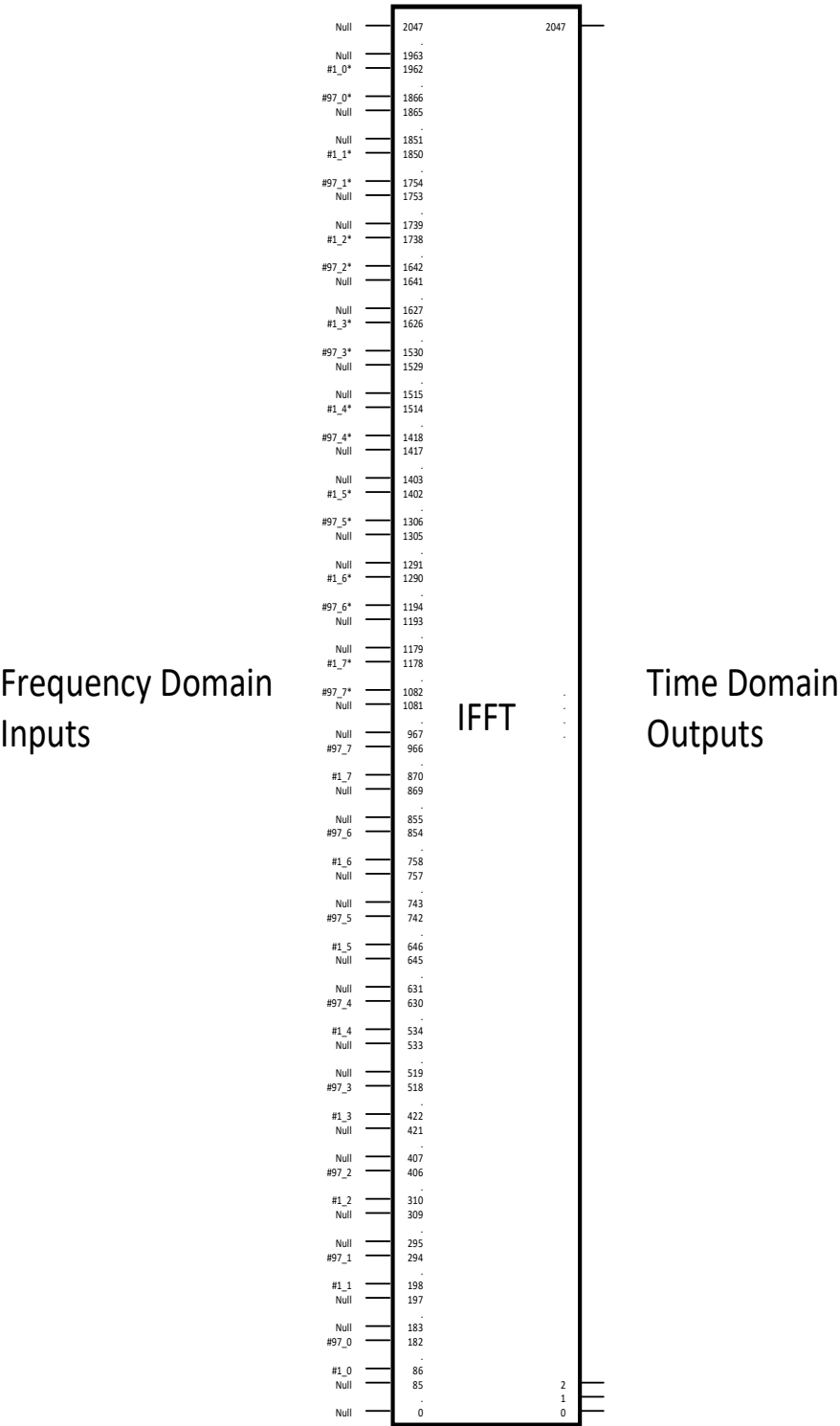
The next equation defines the P-ary DPSK constellation of P phases:

$$s_k = A e^{j\theta_k}$$

Where:

- k is the frequency index representing the k^{th} subcarrier in an OFDM symbol. $k = 1$ corresponds to the phase reference pilot subcarrier.
- s_k is the modulator output (a complex number) for the k^{th} given subcarrier.
- θ_k stands for the absolute phase of the modulated signal, and is obtained as follows:
- $\theta_k = (\theta_{k-1} + (2\pi / P) \Delta b_k) \bmod 2\pi$
- This equation applies for $k > 1$ in the payload, the $k = 1$ subcarrier being the phase reference pilot. When the header is transmitted, the pilot allocated in the k^{th} subcarrier is used as a phase reference for the data allocated in the $(k+1)^{\text{th}}$ subcarrier.
- $\Delta b_k \in \{0, 1, \dots, P-1\}$ represents the information coded in the phase increment, as supplied by the constellation encoder.
- $P = 2, 4$, or 8 in the case of DBPSK, DQPSK or D8PSK, respectively.
- A is a shaping parameter and represents the ring radius from the center of the constellation. The value of A determines the power in each subcarrier and hence the average power transmitted in the header and payload symbols.

492 If a complex 2048-point IFFT is used, the $96 \times N_{CH}$ subcarriers shall be mapped as shown in Figure 18. The
493 symbol * represents complex conjugate.



494

495

496 After the IFFT, the symbol is cyclically extended by 48 samples to create the cyclic prefix (N_{CP}).

3.3.8 Electrical specification of the transmitter

3.3.8.1 General

The following requirements establish the minimum technical transmitter requirements for interoperability, and adequate transmitter performance.

3.3.8.2 Transmit PSD

Transmitter specifications will be measured according to the following conditions and set-up.

For single-phase devices, the measurement shall be taken on either the phase or neutral connection according to Figure 19.

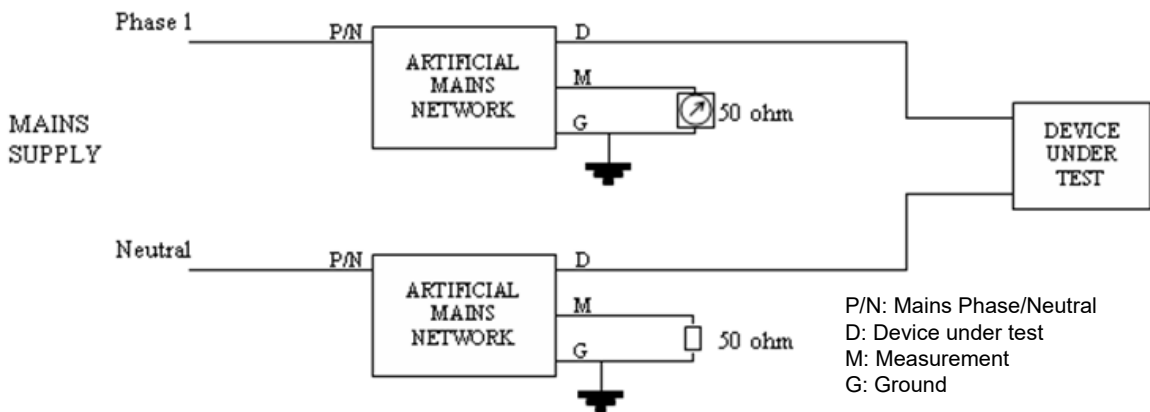


Figure 19 – Measurement set up (single-phase)

For three-phase devices which transmit on all three phases simultaneously, measurements shall be taken in all three phases as per Figure 20. No measurement is required on the neutral conductor.

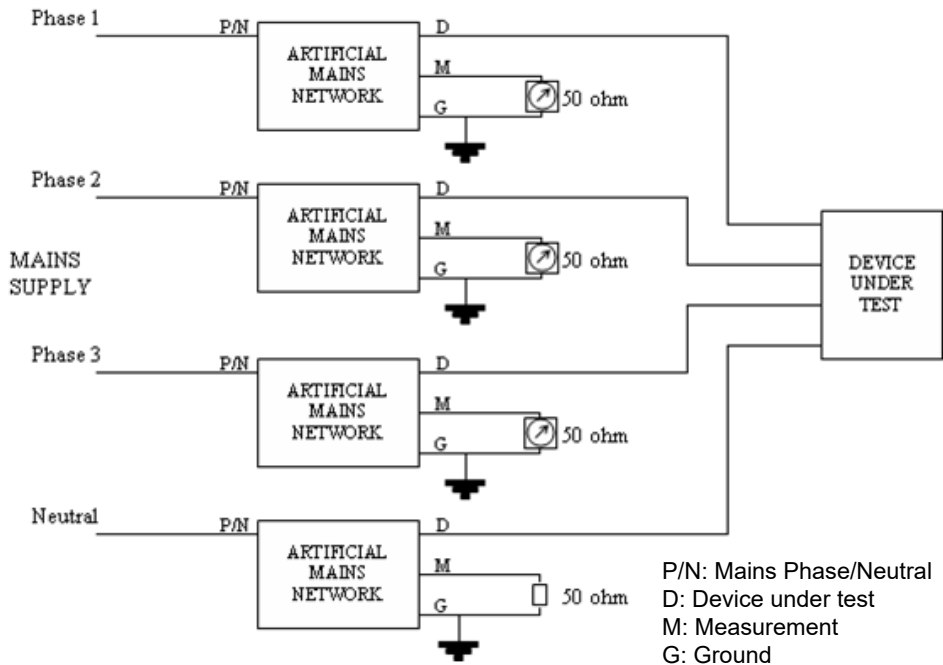


Figure 20 – Measurement set up (three-phase)

The artificial mains network in Figure 19 and Figure 20 is shown in Figure 21. It is based on EN 50065-1:2001. The 33 μ F capacitor and 1 Ω resistor have been introduced so that the network has an impedance of 2 Ω in the frequency band of interest.

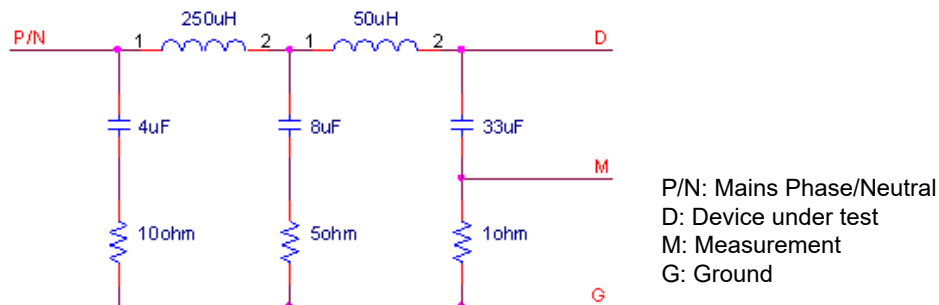


Figure 21 – Artificial mains network

All transmitter output voltages are specified as the voltage measured at the line Terminal with respect to the neutral Terminal. Accordingly, values obtained from the measuring device must be increased by 6 dB (voltage divider of ratio $\frac{1}{2}$).

All devices will be tested to comply with PSD requirements over the full temperature range, which depends on the type of Node:

- Base Nodes in the range -40°C to +70°C
- Service Nodes in the range -25°C to +55°C

All tests shall be carried out under normal traffic load conditions.

In all cases, the PSD must be compliant with the regulations in force in the country where the system is used.

When driving only one phase, the power amplifier shall be capable of injecting a final signal level in the transmission Node (S1 parameter) of 120dB μ Vrms (1 Vrms). This could be in one of two scenarios: either the DUT is connected to a single phase as shown in Figure 19; or the DUT is connected to three phases as shown in Figure 20, but drives only one phase at a time. In both cases, connection is through the AMN of Figure 21.

For three-phase devices injecting simultaneously into all three phases, the final signal level shall be 114dB μ Vrms (0.5Vrms).

Note 1: In all the above cases, note the measurement equipment has some insertion loss. Specifically, in the single-phase, configuration, the measured voltage is 6 dB below the injected signal level, and will equal 114 dB μ V when the injected signal level is 120 dB μ V. Similarly, when connected to three phases, the measured signal level will be 12 dB below the injected signal level. Thus, a 114 dB μ V signal injected into three phases being driven simultaneously, will be measured as 102 dB μ V on any of the three meters of Figure 20.

Note 2: Regional restrictions may apply, ex., on the reactive power drawn from a meter including a PRIME modem. These regulations could affect the powerline interface, and should be accounted for.

3.3.8.3 Error Vector Magnitude (EVM)

The quality of the injected signal with regard to the artificial mains network impedance must be measured in order to validate the transmitter device. Accordingly, a vector analyzer that provides EVM measurements (EVM meter) shall be used, see Annex B for EVM definition. The test set-up described in Figure 19 and Figure 20 shall be used in the case of single-phase devices and three-phase devices transmitting simultaneously on all phases, respectively.

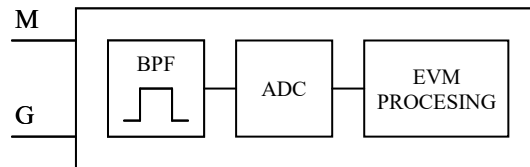


Figure 22 – EVM meter (block diagram)

The EVM meter must include a Band Pass Filter with an attenuation of 40 dB at 50 Hz that ensures anti-aliasing for the ADC. The minimum performance of the ADC is 1MSPS, 14-bit ENOB. The ripple and the group delay of the band pass filter must be accounted for in EVM calculations.

3.3.8.4 Conducted disturbance limits

Regional regulations may apply. For instance, in Europe, transmitters shall comply with the maximum emission levels and spurious emissions defined in EN50065-1:2001 for conducted emissions in AC mains in the bands 3 kHz to 9 kHz and 95 kHz to 30 MHz. European regulations also require that transmitters and receivers shall comply with impedance limits defined in EN50065-7:2001 in the range 3 kHz to 148.5 kHz.

3.4 RF PHY

The RF PHY of PRIME, will be based on some of the general PHY requirements defined on clause 10 of IEEE 802.15.4-2015 [28] and of IEEE 802.15.4v-2017 [29].

The support of PRIME profile of SUN FSK PHY, will be mandatory on PRIME devices supporting RF.

3.4.1 PRIME profile of SUN FSK PHY

The PRIME profile of SUN FSK PHY is given in clause 20 of IEEE 802.15.4-2015 [28] amended by IEEE 802.15.4v-2017 [29], together with the following statements and modifications shown in Table 9.

In this clause, 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 for information only.

N = "Normative": The statements of the reference document shall apply without modifications or remarks.

S = "Selection": The statements of the reference document shall apply with the selections specified.

E = "Extension": The statements of the reference document shall apply with the extensions (modifications and remarks noted under the part title) specified.

N/R = "Not Relevant": The statements of the reference document do not apply. An explanation may be given under the part title.

References to clauses in the "Clause" column refer to the referenced document, while references to clauses/annexes in the "Title and remarks" column refer to this specification unless specifically indicated otherwise.

Table 9 - Selections from clause 20 of [28] amended by [29]

Clause	Title and remarks/modifications	Statement
20.1	Introduction	N
20.2	PPDU format for SUN FSK – The mode switch PPDU shall not be used	S
20.2.1	SHR field format	N
20.2.1.1	Preamble field – The attribute phyFskPreambleLength shall be set to eight	S
20.2.1.2	SFD – The attribute phySunFskSfd shall be set to zero	S
20.2.2	PHR field format – The Mode Switch field shall be set to zero – The FCS Type field shall be set to zero on transmission but on reception it is ignored. As the PRIME MAC format is independent to the IEEE 802.15.4-2015 MAC format, the FCS is not transmitted. – The Data Whitening field shall be set to one	N
20.2.3	Mode Switch PHR – Mode switch PPDUs shall not be used	N/R
20.2.4	PHY Payload field	N
20.3	Modulation and coding for SUN FSK – The 863–870 MHz frequency band shall be supported – Other frequency bands may be supported – Operating Mode #1 shall be supported – Operating Mode #2 may be supported – The operating mode shall be administratively configured	S
20.3.1	Reference modulator	N
20.3.2	Bit-to-symbol mapping	N
20.3.3	Modulation quality	N
20.3.3.1	Frequency deviation tolerance	N
20.3.3.2	Zero crossing tolerance	N
20.3.4	FEC – The FEC mode shall be administratively configured – If FEC is enabled, NRNSC shall be used	S
20.3.5	Code-symbol interleaving – NRNSC shall be supported	S
20.4	Data whitening for SUN FSK	N
20.5	Mode switch mechanism for SUN FSK – Mode switch PPDUs shall not be used	N/R
20.6	SUN FSK PHY RF requirements	N

Table 9 - Selections from clause 20 of [28] amended by [29]

Clause	Title and remarks/modifications	Statement
20.6.1	Operating frequency range	N
20.6.2	Regulatory compliance	N
20.6.3	Radio frequency tolerance – The value of T shall be computed as – $T \leq \min(T0 \times R \times h \times F0 / (R0 \times h0 \times F), 20 \times 10^{-6})$	N
20.6.4	Channel switch time	N
20.6.5	Transmitter symbol rate	N
20.6.6	Transmit spectral mask	N
20.6.7	Receiver sensitivity	N
20.6.8	Receiver interference rejection	N
20.6.9	TX-to-RX turnaround time	N
20.6.10	RX-to-TX turnaround time	N
20.6.11	Transmit power	N
20.6.12	Receiver maximum input level of desired signal	N
20.6.13	Receiver ED	N
20.6.14	LQI - The RF forward LQI shall be measured for each received packet and is based on the received signal power spectral density (PSD). The received signal PSD is a measurement of the received signal power at the antenna at 1 Hz bandwidth. The LQI is 1 byte value and it is mapped to the PSD as follows: • PSD < -174 dBm maps to LQI 0x00 • PSD > 80 dBm maps to LQI 0xFE • -174 dBm ≤ PSD ≤ 80 dBm is linearly interpolated between 0x00 and 0xFE (the nominal step size is 1 dB) The LQI value 0xFF represents a “not measured” LQI. Note: The PSD can be derived from the RSSI subtracting the log of the bandwidth ($10 * \log_{10}(BW)$). As an example, if the signal bandwidth is 100kHz, PSD = RSSI – 50.	N

573

574 3.5 PHY service specification

575 3.5.1 General

576 PHY shall have a single 20-bit free-running clock incremented in steps of 10 μs. The clock counts from 0 to
577 1048575 then overflows back to 0. As a result the period of this clock is 10.48576 seconds. The clock is never
578 stopped nor restarted. Time measured by this clock is the one to be used in some PHY primitives to indicate
579 a specific instant in time.

3.5.2 PHY Data plane primitives

3.5.2.1 General

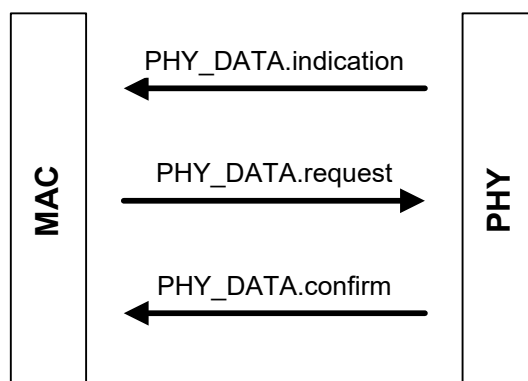


Figure 23 – Overview of PHY primitives

The PHY Abstraction Layer will provide to the MAC Layer all PHY Data plane primitives, to ensure the transmission and reception of MPDUs between Neighbor Nodes using PLC or RF.

The request primitive is passed from MAC to PHY to request the initiation of a service. On the transmission of MPDUs the PHY Abstraction Layer uses PLC PHY or RF PHY depending on the selection of the MAC layer. This selection is communicated from the MAC to the PHY Abstraction Layer using the Medium parameter.

Medium parameter will use the PCH (Physical channel characteristics) coding used also on PRO.PCH field of Table 29 of clause 4.4.2.6.5.1(Extension for Multi-PHY promotion) of this document.

The indication and confirm primitives are passed from PHY to MAC to indicate an internal PHY event that is significant to MAC. This event may be logically related to a remote service request or may be caused by an event internal to PHY.

3.5.2.2 PHY_DATA.request

3.5.2.2.1 Function

The PHY_DATA.request primitive is passed to the PHY layer entity to request the sending of a PPDU to one or more remote PHY entities using the PHY transmission procedures. It also allows setting the time at which the transmission must be started.

3.5.2.2.2 Structure

The semantics of this primitive are as follows:

PHY_DATA.request{MPDU, Length, Level, Type, Scheme, Scheduled, Time, Medium}.

The *MPDU* parameter specifies the MAC protocol data unit to be transmitted by the PHY layer entity. When *Medium* is a PLC PCH, it is mandatory for implementations to byte-align the MPDU across the PHY-SAP. This implies 2 extra bits (due to the non-byte-aligned nature of the MAC layer Header) to be located at the beginning of the header (Type A).

The *Length* parameter specifies the length of MPDU in bytes. Length is 2 bytes long.

607 The *Level* parameter specifies the output signal level according to which the PHY layer transmits MPDU. It
608 may take one of eight values:

609 0: Maximal output level (MOL)

610 1: MOL -3 dB

611 2: MOL -6 dB

612 ...

613 7: MOL -21 dB

614 The *Type* parameter specifies the PHY frame type which should be used for the transmission. When *Medium*
615 is a PLC PCH:

616 0: PHY frame Type A

617 1: PHY frame Type B

618 The *Scheme* parameter specifies the transmission scheme to be used for MPDU.

619 When *Medium* is a PLC PCH, *Scheme* can have any of the following values:

620 0: DBPSK

621 1: DQPSK

622 2: D8PSK

623 3: Not used

624 4: DBPSK + Convolutional Code

625 5: DQPSK + Convolutional Code

626 6: D8PSK + Convolutional Code

627 7-11: Not used

628 12: Robust DBPSK

629 13: Robust DQPSK

630 14-15: Not used

631 If *Scheduled* is false, the transmissions shall start as soon as possible. If *Scheduled* is true, the *Time* parameter
632 is taken into account. The *Time* parameter specifies the instant in time in which the MPDU has to be
633 transmitted. It is expressed in 10s of μ s and may take values from 0 to $2^{20}-1$.

634 The *Medium* is the PCH (Physical channel characteristics) to be used by the PHY on the request.

Note that the Time parameter should be calculated by the MAC, taking into account the current PHY time which may be obtained by PHY_timer.get primitive. The MAC should account for the fact that no part of the PPDU can be transmitted during beacon slots and CFP periods granted to other devices in the network. If the time parameter is set such that these rules are violated, the PHY will return a fail in PHY_Data.confirm.

3.5.2.2.3 Use

The primitive is generated by the MAC layer entity whenever data is to be transmitted to a peer MAC entity or entities.

The MAC layer sends a PHY_DATA.request for each of the mediums for which it wants to send the PPDU. The PHY Abstraction Layer will send PHY_DATA.request to PLC PHY or RF PHY depending on the *Medium* requested by MAC Layer. The reception of this primitive will cause the PHY entity to perform all the PHY-specific actions and pass the properly formed PPDU to the PLC PHY or RF PHY selected on the *Medium* parameter to transfer to the peer PHY layer entity or entities. The next transmission shall start when Time = Timer.

3.5.2.3 PHY_DATA.confirm

3.5.2.3.1 Function

The PHY_DATA.confirm primitive has only local significance and provides an appropriate response to a PHY_DATA.request primitive. The PHY_DATA.confirm primitive tells the MAC layer entity whether or not the MPDU of the previous PHY_DATA.request has been successfully transmitted.

3.5.2.3.2 Structure

The semantics of this primitive are as follows:

PHY_DATA.confirm{*Result*, *Medium*}.

The *Result* parameter is used to pass status information back to the local requesting entity. It is used to indicate the success or failure of the previous associated PHY_DATA.request. Some results will be standard for all implementations:

0: Success.

1: Too late. Time for transmission is past.

2: Invalid *Length*.

3: Invalid *Scheme*.

4: Invalid *Level*.

5: Buffer overrun.

6: Busy channel.

7-255: Proprietary.

The *Medium* is the PCH (Physical channel characteristics) used by the PHY on the request.

3.5.2.3.3 Use

The PHY layer sends to the MAC layer a PHY_DATA.confirm for each of the PHY_DATA.request received from MAC.

It is assumed that the MAC layer has sufficient information to associate the confirm primitive with the corresponding request primitive.

3.5.2.4 PHY_DATA.indication**3.5.2.4.1 Function**

This primitive defines the transfer of data from the PHY layer entity to the MAC layer entity.

3.5.2.4.2 Structure

The semantics of this primitive are as follows:

PHY_DATA.indication{*PSDU, Length, Level, Type, Scheme, Time, Medium*}.

The *PSDU* parameter specifies the PHY service data unit as received by the local PHY layer entity. When *Medium* is a PLC PCH, it is mandatory for implementations to byte-align MPDU across the PHY-SAP. For Type A frames, this implies 2 extra bits (due to the non-byte-aligned nature of the MAC layer Header) to be located at the beginning of the header.

The *Length* parameter specifies the length of received PSDU in bytes. Length is 2 bytes long.

The *Level* parameter specifies the signal level on which the PHY layer received the PSDU.

In case *Medium* is a PLC PHY, *Level* may take one of sixteen values:

0: ≤ 70 dBuV

1: ≤ 72 dBuV

2: ≤ 74 dBuV

...

15: > 98 dBuV

In case *Medium* is SUN FSK PHY, the underlying estimator is the receiver ED defined in 20.6.13 [28], which is a power measurement over the receiver band (signal and noise) and *Level* may take one of sixteen values:

0: ≤ -87 dBm

1: ≤ -82 dBm

2: ≤ -77 dBm

...

15: > -17 dBm

698 The *Type* parameter specifies the PHY frame type with which PSDU is received. When *Medium* is a PLC PCH:

699 0: PHY frame Type A

700 1: PHY frame Type B. The *Scheme* parameter specifies the scheme with which PSDU is received.

701 When *Medium* is a PLC PCH, *Scheme* can have any of the following values:

702 0: DBPSK

703 1: DQPSK

704 2: D8PSK

705 3: Not used

706 4: DBPSK + Convolutional Code

707 5: DQPSK + Convolutional Code

708 6: D8PSK + Convolutional Code

709 7-11: Not used

710 12: Robust DBPSK

711 13: Robust DQPSK

712 14-15: Not used

713 The *Time* parameter is the time of receipt of the Preamble associated with the PSDU.

714 The *Medium* is the PCH (Physical channel characteristics) used by the PHY to receive the PSDU.

715 **3.5.2.4.3 Use**

716 The PHY layer sends to the MAC layer a PHY_DATA.indication for each of the PSDU received from a Medium.

717 **3.5.3 PHY Control plane primitives**

718 **3.5.3.1 General**

719 Figure 24 shows the generate structure of PHY control plane primitives. Each primitive may have "set", "get"
720 and "confirm" fields. Table 10 below lists the control plane primitives and the fields associated with each of
721 them. Each row is a control plane primitive. An "X" in a column indicates that the associated field is used in
722 the primitive described in that row.

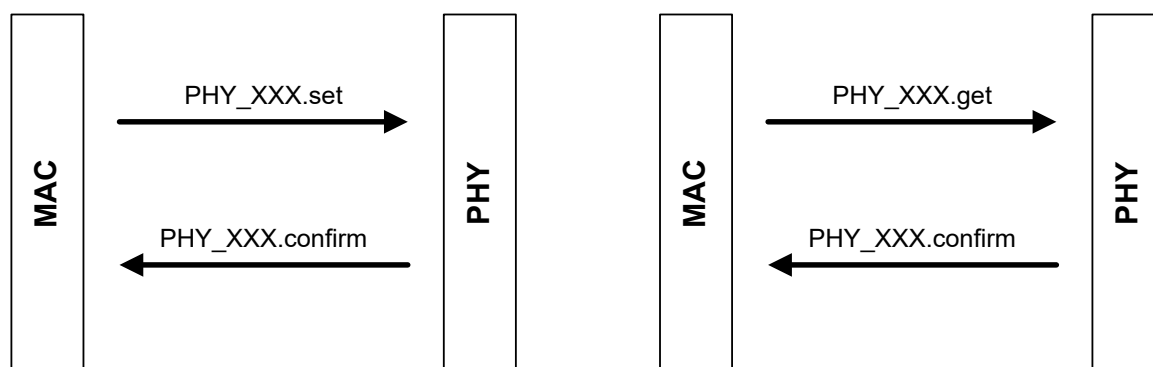


Figure 24 – Overview of PHY Control Plane Primitives

Table 10 - Fields associated with PHY Control Plane Primitives

Field	PLC PHY	RF PHY	set	get	confirm
PHY_AGC	X		X	X	X
PHY_Timer	X	X		X	X
PHY_CD	X			X	X
PHY_CCA		X		X	X
PHY_NL	X	X		X	X
PHY_SNR	X	X		X	X
PHY_ZCT	X			X	X
PHY_CH	X	X	X	X	X

3.5.3.2 PHY_AGC.set

3.5.3.2.1 Function

The PHY_AGC.set primitive is passed to the PHY layer entity by the MAC layer entity to set the Automatic Gain Mode of the PHY layer.

3.5.3.2.2 Structure

The semantics of this primitive are as follows:

PHY_AGC.set {*Mode*, *Gain* }.

The *Mode* parameter specifies whether or not the PHY layer operates in automatic gain mode. It may take one of two values:

0: Auto;

1: Manual.

The *Gain* parameter specifies the initial receiving gain in auto mode. It may take one of N values:

0: *min_gain* dB;

1: *min_gain* + *step* dB;

2: *min_gain* + 2**step* dB;

741 ...

742 N-1: $\text{min\_gain} + (N-1) \cdot \text{step}$ dB.

743 where min_gain and N depend on the specific implementation. step is also an implementation issue but it
744 shall not be more than 6 dB. The maximum Gain value $\text{min\_gain} + (N-1) \cdot \text{step}$ shall be at least 21 dB.

745 3.5.3.2.3 Use

746 The primitive is generated by the MAC layer when the receiving gain mode has to be changed.

747 3.5.3.3 PHY_AGC.get

748 3.5.3.3.1 Function

749 The PHY_AGC.get primitive is passed to the PHY layer entity by the MAC layer entity to get the Automatic
750 Gain Mode of the PHY layer.

751 3.5.3.3.2 Structure

752 The semantics of this primitive are as follows:

753 PHY_AGC.get{ }.

754 3.5.3.3.3 Use

755 The primitive is generated by the MAC layer when it needs to know the receiving gain mode that has been
756 configured.

757 3.5.3.4 PHY_AGC.confirm

758 3.5.3.4.1 Function

759 The PHY_AGC.confirm primitive is passed by the PHY layer entity to the MAC layer entity in response to a
760 PHY_AGC.set or PHY_AGC.get command.

761 3.5.3.4.2 Structure

762 The semantics of this primitive are as follows:

763 PHY_AGC.confirm {*Mode*, *Gain*}.

764 The *Mode* parameter specifies whether or not the PHY layer is configured to operate in automatic gain mode.
765 It may take one of two values:

766 0: Auto;

767 1: Manual.

768 The *Gain* parameter specifies the current receiving gain. It may take one of N values:

769 0: min_gain dB;

770 1: $\text{min\_gain} + \text{step}$ dB;

771 2: $\text{min\_gain} + 2 * \text{step}$ dB;

772 ...

773 N-1: $\text{min\_gain} + (N-1) * \text{step}$ dB.

774 where *min_gain* and N depend on the specific implementation. *step* is also an implementation issue but it
775 shall not be more than 6 dB. The maximum *Gain* value $\text{min\_gain} + (N-1) * \text{step}$ shall be at least 21 dB.

776 3.5.3.5 PHY_Timer.get

777 3.5.3.5.1 Function

778 The PHY_Timer.get primitive is passed to the PHY layer entity by the MAC layer entity to get the current PHY
779 time.

780 3.5.3.5.2 Structure

781 The semantics of this primitive are as follows:

782 PHY_Timer.get {*Medium*}.

783 The *Medium* is the PCH (Physical channel characteristics) used by the PHY to receive the PSDU.

784 3.5.3.5.3 Use

785 The primitive is generated by the MAC layer to know the current PHY time.

786 3.5.3.6 PHY_Timer.confirm

787 3.5.3.6.1 Function

788 The PHY_Timer.confirm primitive is passed to the MAC layer by the PHY layer entity in response to a
789 PHY_Timer.get command.

790 3.5.3.6.2 Structure

791 The semantics of this primitive are as follows:

792 PHY_Timer.confirm {*Time*, *Medium*}.

793 The *Time* parameter is specified in 10s of microseconds. It may take values of between 0 and $2^{20}-1$.

794 The *Medium* is the PCH (Physical channel characteristics) used by the PHY to receive the PSDU.

795 3.5.3.7 PHY_CD.get

796 3.5.3.7.1 Function

797 The PHY_CD.get primitive is passed to the PHY layer entity by the MAC layer entity to look for the carrier
798 detect signal. The carrier detection algorithm shall be based on preamble detection and header recognition
799 (see Section 3.3.2).

3.5.3.7.2 Structure

The semantics of this primitive are as follows:

PHY_CD.get { }.

3.5.3.7.3 Use

The primitive is generated by the MAC layer when it needs to know whether or not the physical medium is free.

3.5.3.8 PHY_CD.confirm**3.5.3.8.1 Function**

The PHY_CD.confirm primitive is passed to the MAC layer entity by the PHY layer entity in response to a PHY_CD.get command.

3.5.3.8.2 Structure

The semantics of this primitive are as follows:

PHY_CD.confirm {*cd*, *rssi*, *Time*, *header*}.

The *cd* parameter may take one of two values:

0: no carrier detected;

1: carrier detected.

The *rssi* parameter is the Received Signal Strength Indication, not including the noise power. One of the RSSI estimator examples is shown in Annex B, but it is implementation specific. It is only relevant when *cd* equals 1. It may take one of sixteen values:

0: ≤ 70 dBuV;

1: ≤ 72 dBuV;

2: ≤ 74 dBuV;

...

15: > 98 dBuV. The *Time* parameter indicates the instant at which the present PPDU will finish. It is only relevant when *cd* equals 1. When *cd* equals 0, *Time* parameter will take a value of 0. If *cd* equals 1 but the duration of the whole PPDU is still not known (i.e. the header has not yet been processed), *header* parameter will take a value of 1 and *time* parameter will indicate the instant at which the header will finish, specified in 10s of microseconds. In any other case the value of *Time* parameter is the instant at which the present PPDU will finish, and it is specified in 10s of microseconds. *Time* parameter refers to an absolute point in time so it is referred to the system clock.

The *header* parameter may take one of two values:

831 1: if a preamble has been detected but the duration of the whole PPDU is not yet known from
832 decoding the header;

833 0: in any other case.

834 **3.5.3.9 PHY_NL.get**

835 **3.5.3.9.1 Function**

836 The PHY_NL.get primitive is passed to the PHY layer entity by the MAC layer to get the noise floor level value.
837 One of the noise estimator examples is shown in Annex B, but it is implementation specific.

838 **3.5.3.9.2 Structure**

839 The semantics of this primitive are as follows:

840 PHY_NL.get {*Medium*}.

841 The *Medium* is the PCH (Physical channel characteristics) used by the PHY to receive the PSDU.

842 **3.5.3.9.3 Use**

843 The primitive is generated by the MAC layer when it needs to know the noise level present in the powerline.

844 **3.5.3.10 PHY_NL.confirm**

845 **3.5.3.10.1 Function**

846 The PHY_NL.confirm primitive is passed to the MAC layer entity by the PHY layer entity in response to a
847 PHY_NL.get command.

848 **3.5.3.10.2 Structure**

849 The semantics of this primitive are as follows:

850 PHY_NL.confirm {*noise*, *Medium*}.

851 When *Medium* is a PLC PCH , the *noise* parameter may take one of sixteen values:

852 0: ≤ 50 dBuV;

853 1: ≤ 53 dBuV;

854 2: ≤ 56 dBuV;

855 ...

856 15: > 92 dBuV.

857 When *Medium* is a SUN FSK PCH , the *noise* parameter may take one of sixteen values:

858 0: ≤ -107 dBm

859 1: ≤ -104 dBm

860 2: ≤ -101 dBm

861 ...

862 15: > -65 dBm

863 The *Medium* is the PCH (Physical channel characteristics) used by the PHY to receive the PSDU.

864 **3.5.3.11 PHY_SNR.get**

865 **3.5.3.11.1 Function**

866 The PHY_SNR.get primitive is passed to the PHY layer entity by the MAC layer entity to get the value of the
867 Signal to Noise Ratio, defined as the ratio of measured received signal level to noise level of last received
868 PPDU. The calculation of the SNR is described in Annex B.

869 **3.5.3.11.2 Structure**

870 The semantics of this primitive are as follows:

871 PHY_SNR.get {*Medium*}.

872 The *Medium* is the PCH (Physical channel characteristics) used by the PHY to receive the PSDU.

873 **3.5.3.11.3 Use**

874 The primitive is generated by the MAC layer when it needs to know the SNR in order to analyze channel
875 characteristics and invoke robustness management procedures, if required.

876 **3.5.3.12 PHY_SNR.confirm**

877 **3.5.3.12.1 Function**

878 The PHY_SNR.confirm primitive is passed to the MAC layer entity by the PHY layer entity in response to a
879 PHY_SNR.get command.

880 **3.5.3.12.2 Structure**

881 The semantics of this primitive are as follows: PHY_SNR.confirm{*SNR*, *Medium*}.

882 The *SNR* parameter refers to the Signal to Noise Ratio, defined as the ratio of measured received signal level
883 to noise level of last received PPDU. It may take one of eight values.

884 The mapping of the 3-bit index to the actual SNR value is given below:

885 0: ≤ 0 dB;

886 1: ≤ 3 dB;

887 2: ≤ 6 dB;

888 ...

889 7: > 18 dB.

890 The *Medium* is the PCH (Physical channel characteristics) used by the PHY to receive the PSDU.

891 When *Medium* is PLC, the SNR is calculated as specified in Annex B.

892 **3.5.3.13 PHY_ZCT.get**

893 **3.5.3.13.1 Function**

894 The PHY_ZCT.get primitive is passed to the PHY layer entity by the MAC layer entity to get the zero cross time
895 of the mains and the time between the last transmission or reception and the zero cross of the mains.

896 **3.5.3.13.2 Structure**

897 The semantics of this primitive are as follows:

898 PHY_ZCT.get { }.

899 **3.5.3.13.3 Use**

900 The primitive is generated by the MAC layer when it needs to know the zero cross time of the mains, e.g. in
901 order to calculate the phase to which the Node is connected.

902 **3.5.3.14 PHY_ZCT.confirm**

903 **3.5.3.14.1 Function**

904 The PHY_ZCT.confirm primitive is passed to the MAC layer entity by the PHY layer entity in response to a
905 PHY_ZCT.get command.

906 **3.5.3.14.2 Structure**

907 The semantics of this primitive are as follows:

908 PHY_ZCT.confirm {*Time* }.

909 The *Time* parameter is the instant in time at which the last zero-cross event took place.

910 **3.5.3.15 PHY_CCA.get**

911 **3.5.3.15.1 Function**

912 The PHY_CCA.get primitive is passed to the PHY layer entity by the MAC layer entity for making Clear Channel
913 assessment mode 1, as described in 10.2.7 of [28]. The PHY PIBs phyCCADuration and phyCCAThreshold
914 control this service.

915 **3.5.3.15.2 Structure**

916 The semantics of this primitive are as follows:

917 PHY_CCA.get { }.

918 **3.5.3.15.3 Use**

919 The primitive is generated by the MAC layer when it needs to know whether or not the RF physical medium
920 is free.

921 3.5.3.16 PHY_CCA.confirm**922 3.5.3.16.1 Function**

923 The PHY_CCA.confirm primitive is passed to the MAC layer entity by the PHY layer entity in response to a
924 PHY_CCA.get command.

925 3.5.3.16.2 Structure

926 The semantics of this primitive are as follows:

927 PHY_CCA.confirm {*channelState*}.

928 The *channelState* parameter may take one of two values:

929 0: channel busy

930 1: channel free

931 3.5.3.17 PHY_CH.set**932 3.5.3.17.1 Function**

933 The PHY_CH.set primitive is passed to the PHY layer entity by the MAC layer entity to set the band (PLC) or
934 the channel (RF) of the PHY medium.

935 3.5.3.17.2 Structure

936 The semantics of this primitive are as follows:

937 PHY_CH.set { *Medium* }.

938 The *Medium* parameter has the format of the PCH (Physical channel characteristics) of 4.4.2.6.5.1..

939 3.5.3.17.3 Use

940 The primitive is generated by the MAC layer in order to set the band (PLC) or the channel (RF).

941 3.5.3.18 PHY_CH.get**942 3.5.3.18.1 Function**

943 The PHY_CH.get primitive is passed to the PHY layer entity by the MAC layer entity to get the band (PLC) or
944 the channel (RF) of a PHY medium.

945 3.5.3.18.2 Structure

946 The semantics of this primitive are as follows:

947 PHY_CH.get{*Medium*}.

948 The *Medium* parameter has the format of the PCH (Physical channel characteristics) of 4.4.2.6.5.1 .

949 3.5.3.18.3 Use

950 The primitive is generated by the MAC layer when it needs to know the current band (PLC) or channel (RF)
951 configured in the related physical medium..

952 3.5.3.19 PHY_CH.confirm**953 3.5.3.19.1 Function**

954 The PHY_CH.confirm primitive is passed by the PHY layer entity to the MAC layer entity in response to a
955 PHY_CH.set or PHY_CH.get command.

956 3.5.3.19.2 Structure

957 The semantics of this primitive are as follows:

958 PHY_CH.confirm { *Medium* }.

959 .

960 The *Medium parameter has the format of* the PCH (Physical channel characteristics) of 4.4.2.6.5.1.

961 3.5.4 PHY Management primitives**962 3.5.4.1 General**

963 PHY layer management primitives enable the conceptual PHY layer management entity to interface to upper
964 layer management entities. Implementation of these primitives is optional. Please refer to Figure 24 to see
965 the general structure of the PHY layer management primitives.

966 3.5.4.2 PLME_RESET.request**967 3.5.4.2.1 Function**

968 The PLME_RESET.request primitive is invoked to request the PHY layer to reset its present functional state.
969 As a result of this primitive, the PHY should reset all internal states and flush all buffers to clear any queued
970 receive or transmit data. All the SET primitives are invoked by the PLME, and addressed to the PHY to set
971 parameters in the PHY. The GET primitive is also sourced by the PLME, but is used only to read PHY
972 parameters

973 3.5.4.2.2 Structure

974 The semantics of this primitive are as follows:

975 PLME_RESET.request{ *Medium* }.

976 The *Medium* is the PCH (Physical channel characteristics) used by the PHY to receive the PSDU.

977 3.5.4.2.3 Use

978 The upper layer management entities will invoke this primitive to tackle any system level anomalies that
979 require aborting any queued transmissions and restart all operations from initialization state.

980 3.5.4.3 PLME_RESET.confirm**981 3.5.4.3.1 Function**

982 The PLME_RESET.confirm is generated in response to a corresponding PLME_RESET.request primitive. It
983 provides indication if the requested reset was performed successfully or not.

984 3.5.4.3.2 Structure

985 The semantics of this primitive are as follows:

986 PLME_RESET.confirm{*Result*, *Medium* }.

987 The *Result* parameter shall have one of the following values:

988 0: Success;

989 1: Failure. The requested reset failed due to internal implementation issues.

990 The *Medium* is the PCH (Physical channel characteristics) used by the PHY to receive the PSDU.

991 3.5.4.3.3 Use

992 The primitive is generated in response to a PLME_RESET.request.

993 3.5.4.4 PLME_SLEEP.request**994 3.5.4.4.1 Function**

995 The PLME_SLEEP.request primitive is invoked to request the PHY layer to suspend its present activities
996 including all reception functions. The PHY layer should complete any pending transmission before entering
997 into a sleep state.

998 3.5.4.4.2 Structure

999 The semantics of this primitive are as follows:

1000 PLME_SLEEP.request{ *Medium* }.

1001 The *Medium* is the PCH (Physical channel characteristics) used by the PHY to receive the PSDU.

1002 3.5.4.4.3 Use

1003 Although this specification pertains to communication over power lines, it may still be objective of some
1004 applications to optimize their power consumption. This primitive is designed to help those applications
1005 achieve this objective.

1006 3.5.4.5 PLME_SLEEP.confirm**1007 3.5.4.5.1 Function**

1008 The PLME_SLEEP.confirm is generated in response to a corresponding PLME_SLEEP.request primitive and
1009 provides information if the requested sleep state has been entered successfully or not.

1010 3.5.4.5.2 Structure

1011 The semantics of this primitive are as follows:

1012 PLME_SLEEP.confirm{*Result*, *Medium*}.

1013 The *Result* parameter shall have one of the following values:

1014 0: Success;

1015 1: Failure. The requested sleep failed due to internal implementation issues;

1016 2: PHY layer is already in sleep state.

1017 The *Medium* is the PCH (Physical channel characteristics) used by the PHY to receive the PSDU.

1018 3.5.4.5.3 Use

1019 The primitive is generated in response to a PLME_SLEEP.request

1020 3.5.4.6 PLME_RESUME.request**1021 3.5.4.6.1 Function**

1022 The PLME_RESUME.request primitive is invoked to request the PHY layer to resume its suspended activities.

1023 As a result of this primitive, the PHY layer shall start its normal transmission and reception functions.

1024 3.5.4.6.2 Structure

1025 The semantics of this primitive are as follows:

1026 PLME_RESUME.request{ *Medium* }.

1027 The *Medium* is the PCH (Physical channel characteristics) used by the PHY to receive the PSDU.

1028 3.5.4.6.3 Use

1029 This primitive is invoked by upper layer management entities to resume normal PHY layer operations,
1030 assuming that the PHY layer is presently in a suspended state as a result of previous PLME_SLEEP.request
1031 primitive.

1032 3.5.4.7 PLME_RESUME.confirm**1033 3.5.4.7.1 Function**

1034 The PLME_RESUME.confirm is generated in response to a corresponding PLME_RESUME.request primitive
1035 and provides information about the requested resumption status.

1036 3.5.4.7.2 Structure

1037 The semantics of this primitive are as follows:

1038 PLME_RESUME.confirm{*Result*, *Medium*}.

1039 The *Result* parameter shall have one of the following values:

- 1040 0: Success;
- 1041 1: Failure. The requested resume failed due to internal implementation issues;
- 1042 2: PHY layer is already in fully functional state.
- 1043 The *Medium* is the PCH (Physical channel characteristics) used by the PHY to receive the PSDU.
- 1044 **3.5.4.7.3 Use**
- 1045 The primitive is generated in response to a PLME_RESUME.request
- 1046 **3.5.4.8 PLME_TESTMODE.request**
- 1047 **3.5.4.8.1 Function**
- 1048 The PLME_TESTMODE.request primitive is invoked to enter the PHY layer to a test mode (specified by the
- 1049 mode parameter). A valid packet is transmitted and the PSDU will contain a defined reference: dummy 54-
- 1050 bit MAC header, message "PRIME IS A WONDERFUL TECHNOLOGY" (note the blank spaces so it represents
- 1051 240 uncoded bits in ASCII format) concatenated as many times as needed to make it 256bytes. The last eight
- 1052 bits will be substituted for eight flushing bits set to zero. Following receipt of this primitive, the PHY layer
- 1053 should complete any pending transmissions in its buffer before entering the requested Test mode..
- 1054 **3.5.4.8.2 Structure**
- 1055 The semantics of this primitive are as follows:
- 1056 PLME_TESTMODE.request{*enable, mode, modulation, pwr_level, Medium* }.
- 1057 The *enable* parameter starts or stops the Test mode and may take one of two values:
- 1058 0: stop test mode and return to normal functional state;
- 1059 1: transit from present functional state to Test mode.
- 1060 The *mode* parameter enumerates specific functional behavior to be exhibited while the PHY is in Test mode.
- 1061 It may have either of the two values.
- 1062 0: continuous transmit;
- 1063 1: transmit with 50% duty cycle.
- 1064 The *modulation* parameter specifies which modulation scheme is used during transmissions. It may take any
- 1065 of the following values:
- 1066 0: DBPSK;
- 1067 1: DQPSK;
- 1068 2: D8PSK;
- 1069 3: Not used;

1070	4: DBPSK + Convolutional Code;
1071	5: DQPSK + Convolutional Code;
1072	6: D8PSK + Convolutional Code;
1073	7-11: Not used
1074	12: Robust DBPSK
1075	13: Robust DQPSK
1076	14-15: Not used
1077	The <i>pwr_level</i> parameter specifies the relative level at which the test signal is transmitted. It may take either of the following values:
1078	
1079	0: Maximal output level (MOL);
1080	1: MOL -3 dB;
1081	2: MOL -6 dB;
1082	...
1083	7: MOL -21 dB;
1084	The <i>Medium</i> is the PCH (Physical channel characteristics) used by the PHY to receive the PSDU.
1085	3.5.4.8.3 Use
1086	This primitive is invoked by management entity when specific tests are required to be performed.
1087	3.5.4.9 PLME_TESTMODE.confirm
1088	3.5.4.9.1 Function
1089	The PLME_TESTMODE.confirm is generated in response to a corresponding PLME_TESTMODE.request primitive to indicate if transition to Testmode was successful or not.
1090	
1091	3.5.4.9.2 Structure
1092	The semantics of this primitive are as follows:
1093	PLME_TESTMODE.confirm{ <i>Result</i> , <i>Medium</i> }.
1094	The <i>Result</i> parameter shall have one of the following values:
1095	0: Success;
1096	1: Failure. Transition to Testmode failed due to internal implementation issues;
1097	2: PHY layer is already in Testmode.

1098 The *Medium* is the PCH (Physical channel characteristics) used by the PHY to receive the PSDU.

1099 **3.5.4.9.3 Use**

1100 The primitive is generated in response to a PLME_TESTMODE.request

1101 **3.5.4.10 PLME_GET.request**

1102 **3.5.4.10.1 Function**

1103 The PLME_GET.request queries information about a given PIB attribute.

1104 **3.5.4.10.2 Structure**

1105 The semantics of this primitive is as follows:

1106 PLME_GET.request{PIBAttribute }

1107 The *PIBAttribute* parameter identifies specific attribute as enumerated in *Id* fields of tables that enumerate
1108 PIB attributes (Section 6.2.2).

1109 **3.5.4.10.3 Use**

1110 This primitive is invoked by the management entity to query one of the available PIB attributes.

1111 **3.5.4.11 PLME_GET.confirm**

1112 **3.5.4.11.1 Function**

1113 The PLME_GET.confirm primitive is generated in response to the corresponding PLME_GET.request
1114 primitive.

1115 **3.5.4.11.2 Structure**

1116 The semantics of this primitive is as follows:

1117 PLME_GET.confirm{status, PIBAttribute, PIBAttributeValue }

1118 The *status* parameter reports the result of requested information and may have one of the values shown in
1119 Table 11.

1120 **Table 11 - Values of the status parameter in PLME_GET.confirm primitive**

Result	Description
<i>Done</i> = 0	Parameter read successfully
<i>Failed</i> =1	Parameter read failed due to internal implementation reasons.
<i>BadAttr</i> =11	Specified <i>PIBAttribute</i> is not supported

1121

1122 The *PIBAttribute* parameter identifies specific attribute as enumerated in *Id* fields of tables that enumerate
1123 PIB attributes (Section 6.2.2).

1124 The *PIBAttributeValue* parameter specifies the value associated with given *PIBAttribute*.

1125 **3.5.4.11.3 Use**

1126 This primitive is generated by PHY layer in response to a PLME_GET.request primitive.

1127 **3.5.4.12 PLME_SET.request**

1128 **3.5.4.12.1 Function**

1129 The PLME_SET.request requests to configure a PIBAttribute.

1130 **3.5.4.12.2 Structure**

1131 The semantic of the primitive is as follows:

1132 *PLME_SET.request{PIBAttribute, PIBAttributeValue}*

1133 The *PIBAttribute* parameter identifies a specific attribute as enumerated in the *Id* fields of tables that list PIB
1134 attributes (Section 6.2.2).

1135 The *PIBAttributeValue* parameter specifies the value associated with given *PIBAttribute*.

1136 **3.5.4.12.3 Use**

1137 This primitive is invoked by the management entity to set one of the available PIB attributes.

1138 **3.5.4.13 PLME_SET.confirm**

1139 **3.5.4.13.1 Function**

1140 The PLME_SET.confirm primitive is generated in response to the corresponding PLME_SET.request primitive.

1141 **3.5.4.13.2 Structure**

1142 The semantic of this primitive is as follows:

1143 *PLME_SET.confirm{result}*

1144 The *result* parameter may have one of the values given in Table 12.

1145 **Table 12 - Values of the result parameter in the PLME_SET.confirm primitive**

<i>Result</i>	<i>Description</i>
<i>Done = 0</i>	Given value successfully set for specified attribute.
<i>Failed = 1</i>	Failed to set the given value for specified attribute.
<i>BadAttr=11</i>	Requested <i>PIBAttribute</i> is not supported.
<i>OutOfRange=12</i>	Requested <i>PIBAttributeValue</i> is out of range.
<i>ReadOnly=13</i>	Requested PIBAttribute is read only.

1146

4 MAC layer

4.1 Overview

A Subnetwork can be logically seen as a tree structure with two types of Nodes: the Base Node and Service Nodes.

- **Base Node:** It is at the root of the tree structure and it acts as a master Node that provides all Subnetwork elements with connectivity. It manages the Subnetwork resources and connections. There is only one Base Node in a Subnetwork. The Base Node is initially the Subnetwork itself, and any other Node should follow a Registration process to enroll itself on the Subnetwork.
- **Service Node:** They are either leaves or branch points of the tree structure. They are initially in a Disconnected functional state and follow the Registration process in 4.6.1 to become part of the Subnetwork. Service Nodes have two functions in the Subnetwork: keeping connectivity to the Subnetwork for their Application layers, and switching other Nodes' data to propagate connectivity. Devices elements that exhibit Base Node functionality continue to do so as long as they are not explicitly reconfigured by mechanisms that are beyond the scope of this specification. Service Nodes, on the other hand, change their behavior dynamically from "Terminal" functions to "Switch" functions and vice-versa. The changing of functional states occurs in response to certain pre-defined events on the network. Figure 25 shows the functional state transition diagram of a Service Node.

The three functional states of a Service Node are **Disconnected**, **Terminal** and **Switch**:

- **Disconnected:** This is the initial functional state for all Service Nodes. When Disconnected, a Service Node is not able to communicate data or switch other Nodes' data; its main function is to search for a Subnetwork within its reach and try to register on it.
- **Terminal:** When in this functional state a Service Node is able to establish connections and communicate data, but it is not able to switch other Nodes' data.
- **Switch:** When in this functional state a Service Node is able to perform all Terminal functions. Additionally, it is able to forward data to and from other Nodes in the same Subnetwork. It is a branch point on the tree structure.

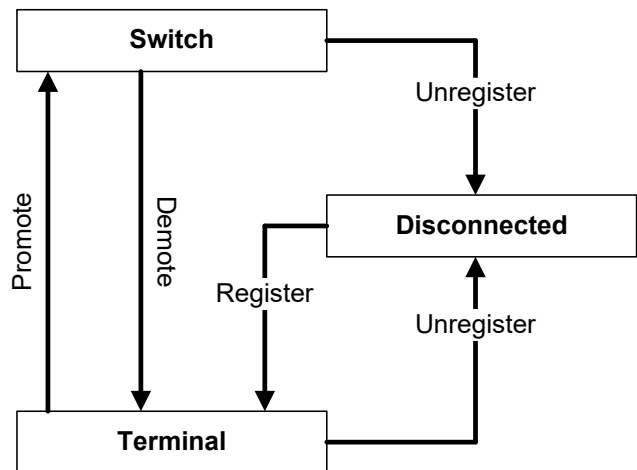


Figure 25 - Service Node states

The events and associated processes that trigger changes from one functional state to another are:

- **Registration:** the process by which a Service Node includes itself in the Base Node's list of registered Nodes. Its successful completion means that the Service Node is part of a Subnetwork. Thus, it represents the transition between Disconnected and Terminal.
- **Unregistration:** the process by which a Service Node removes itself from the Base Node's list of registered Nodes. Unregistration may be initiated by either of Service Node or Base Node. A Service Node may unregister itself to find a better point of attachment i.e. change Switch Node through which it is attached to the network. A Base Node may unregister a registered Service Node as a result of failure of any of the MAC procedures. Its successful completion means that the Service Node is Disconnected and no longer part of a Subnetwork;
- **Promotion:** the process by which a Service Node is qualified to switch (repeat, forward) data traffic from other Nodes and act as a branch point on the Subnetwork tree structure. A successful promotion represents the transition between Terminal and Switch. For nodes that support Multi-PHY promotion (see 4.6.3.3), this process may also apply to a Service Node that it is already a Switch. In this case the node which is switch on one medium becomes switch on both media. When a Service Node is Disconnected it cannot directly transition to Switch;
- **Demotion:** the process by which a Service Node ceases to be a branch point on the Subnetwork tree structure. A successful demotion represents the transition between Switch and Terminal. For nodes that support Multi-PHY promotion (see 4.6.3.3), this process may also apply to a Service Node that is Switch on two media. In this case the node which is a Switch on both media is demoted on one medium and remains Switch in the other medium (see 4.6.4).

A PLC-only Node is a Node that is capable of communicating only over the PLC medium. A RF-only Node is a Node that is capable of communicating only over the RF medium. A PLC+RF Node is a Node that is capable of communicating both over the PLC and RF media. A Subnetwork may include PLC-only Nodes, RF-only Nodes and/or PLC+RF Nodes. A PLC-only Node can be part of a Subnetwork if it has a peer communication with a PLC-only Node (Base Node or Switch) or with a PLC+RF Node (Base Node or Switch). A RF-only Node can be part of a Subnetwork if it has a peer communication with a RF-only Node (Base Node or Switch) or with a PLC+RF Node (Base Node or Switch). A PLC+RF Node can be part of a Subnetwork if it has a peer communication with the Base Node or with a Switch.

A Service Node can be registered to the Subnetwork through only one medium (PLC or RF).

A PLC-only Base Node or PLC-only Switch uses the same PLC band for beacon transmission, uplink and downlink communication. A RF-only Base Node or RF-only Switch uses the same RF channel for beacon transmission, uplink and downlink communication. A PLC+RF Base Node or Switch may use one or two medium: in a medium it uses the same PLC band or the same RF channel for beacon transmission, uplink and downlink communication.

A special use of the RF channels occurs in the case of channel hopping (see 4.6.11).

In Figure 26, a Subnetwork example is reported. BN is the Base Node. It is a PLC+RF node as well as Switch nodes B, C and J (grey nodes). Terminal Nodes A, F, G, H, K, M and Switch Nodes D and L are PLC-only nodes (white nodes). Terminal node I and Switch node E are RF-only nodes (black nodes). PLC communications (continuous lines) occur on the following bands: Channel 6 in the paths BN-A and BN-B-D-H, Channel 7 in the

paths C-F and C-G and Channel 1 in the paths J-K and J-L-M. RF communications (dotted lines) occur on the following channels: Channel 1 in the path BN-J, Channel 2 in the paths B-C and B-E-I.

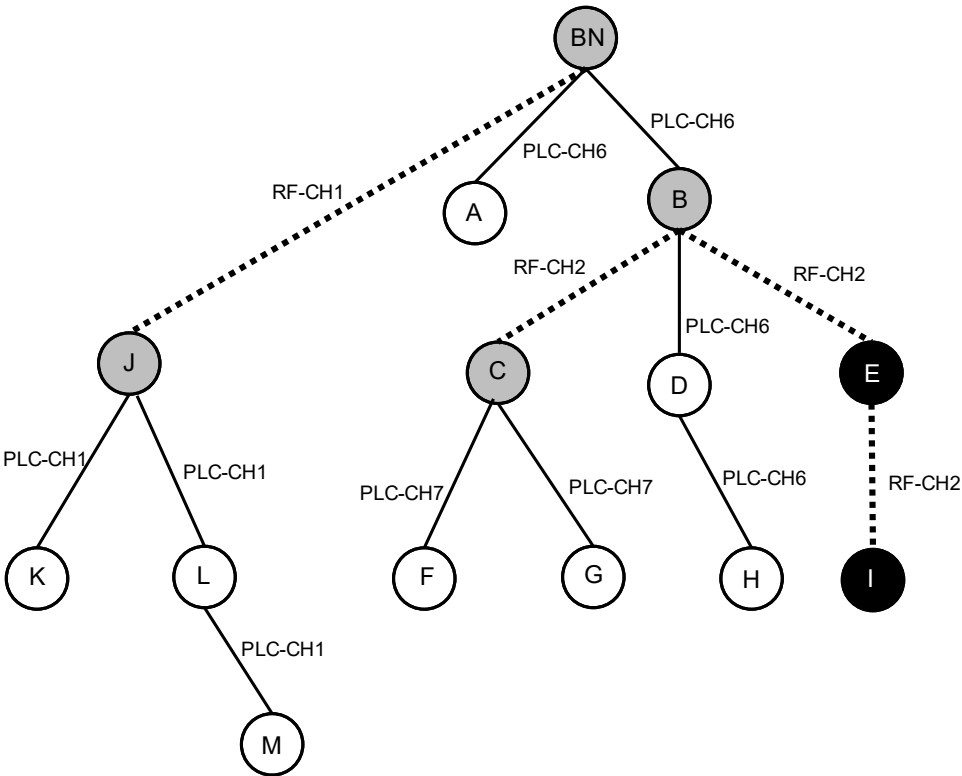


Figure 26 - Subnetwork example

4.2 Addressing

4.2.1 General

Each Node has a 48-bit universal MAC address, defined in IEEE Std 802-2001 and called EUI-48. Every EUI-48 is assigned during the manufacturing process and it is used to uniquely identify a Node during the Registration process.

The EUI-48 of the Base Node uniquely identifies its Subnetwork. This EUI-48 is called the Subnetwork Address (SNA).

The Switch Identifier (LSID) is a unique 8-bit identifier for each Switch Node inside a Subnetwork. The Subnetwork Base Node assigns an LSID during the promotion process. A Switch Node is universally identified by the SNA and LSID. LSID = 0x00 is reserved for the Base Node. LSID = 0xFF is reserved to mean “unassigned” or “invalid” in certain specific fields (see

Table 27). This special use of the 0xFF value is always made explicit when describing those fields and it shall not be used in any other field.

During its Registration process, every Service Node receives a 14-bit Local Node Identifier (LNID). The LNID identifies a single Service Node among all Service Nodes that directly depend on a given Switch. The

combination of a Service Node’s LNID and SID (its immediate Switch’s LSID) forms a 22-bit Node Identifier (NID). The NID identifies a single Service Node in a given Subnetwork. LNID = 0x0000 cannot be assigned to a Terminal, as it refers to its immediate Switch. LNID = 0x3FFF is reserved for broadcast and multicast traffic (see section 4.2.3 for more information). In certain specific fields, the LNID = 0x3FFF may also be used as “unassigned” or “invalid” (see Table 13 and

Table 23). This special use of the 0x3FFF value is always made explicit when describing the said fields and it shall not be used in this way in any other field.

During connection establishment a 9-bit Local Connection Identifier (LCID) is reserved. The LCID identifies a single connection in a Node. The combination of NID and LCID forms a 31-bit Connection Identifier (CID). The CID identifies a single connection in a given Subnetwork. Any connection is universally identified by the SNA and CID. LCID values are allocated with the following rules:

LCID=0x000 to 0x0FF, for connections requested by the Base Node. The allocation shall be made by the Base Node.

LCID=0x100 to 0x1FF, for connections requested by a Service Node. The allocation shall be made by a Service Node.

The full addressing structure and field lengths are shown in Figure 27

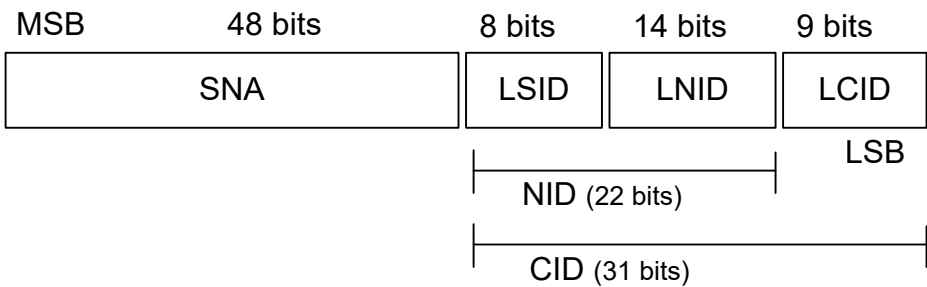


Figure 27 - Addressing Structure

When a Service Node in *Terminal* state starts promotion process, the Base Node allocates a unique switch identifier which is used by this device after transition to switch state as SID of this switch. The promoted Service Node continues to use the same NID that it used before promotion i.e. it maintains SID of its next level switch for addressing all traffic generated/destined to its local application processes. To maintain distinction between the two switch identifiers, the switch identifier allocated to a Service Node during its promotion is referred to as Local Switch Identifier (LSID). Note that the LSID of a switch device will be SID of devices that connects to the Subnetwork through it.

Each Service Node has a level in the topology tree structure. Service Nodes which are directly connected to the Base Node have level 0. The level of any Service Node not directly connected to the Base Node is the level of its immediate Switch plus one.

4.2.2 Example of address resolution

Figure 28 shows an example where Disconnected Service Nodes are trying to register on the Base Node. In this example, addressing will have the following nomenclature: (SID, LNID). Initially, the only Node with an address is Base Node A, which has an NID=(0, 0).

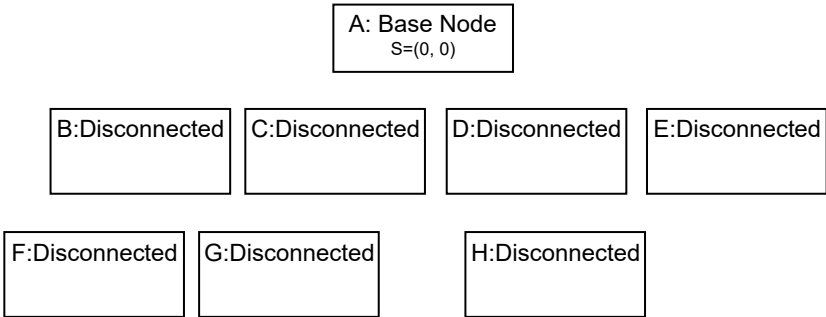


Figure 28 – Example of address resolution: phase 1

Every other Node of the Subnetwork will try to register on the Base Node. Only B, C, D and E Nodes are able to register on this Subnetwork and get their NIDs. Figure 29 shows the status of Nodes after the Registration process. Since they have registered on the Base Node, they get the SID of the Base Node and a unique LNID. The level of newly registered Nodes is 0 because they are connected directly to the Base Node.

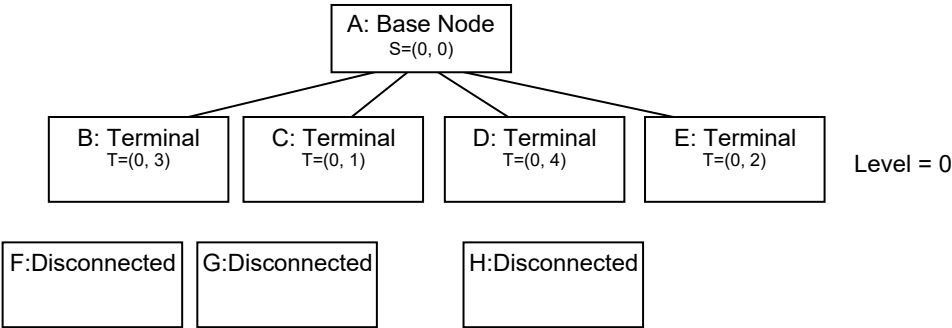


Figure 29 – Example of address resolution: phase 2

Nodes F, G and H cannot connect directly to the Base Node, which is currently the only Switch in the Subnetwork. F, G and H will send PNPDU broadcast requests, which will result in Nodes B and D requesting promotion for themselves in order to extend the Subnetwork range. During promotion, they will both be assigned unique SIDs. Figure 30 shows the new status of the network after the promotion of Nodes B and D. Each Switch Node will still use the NID that was assigned to it during the Registration process for its own communication as a Terminal Node. The new SID shall be used for all switching functions.

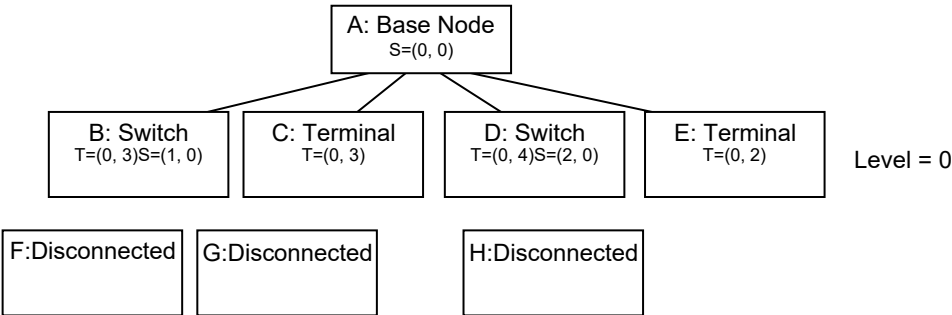


Figure 30 – Example of address resolution: phase 3

On completion of the B and D promotion process, Nodes F, G and H shall start their Registration process and have a unique LNID assigned. Every Node on the Subnetwork will then have a unique NID to communicate like a Terminal, and Switch Nodes will have unique SIDs for switching purposes. The level of newly registered Nodes is 1 because they register with level 0 Nodes. On the completion of topology resolution and address allocation, the example Subnetwork would be as shown in Figure 31.

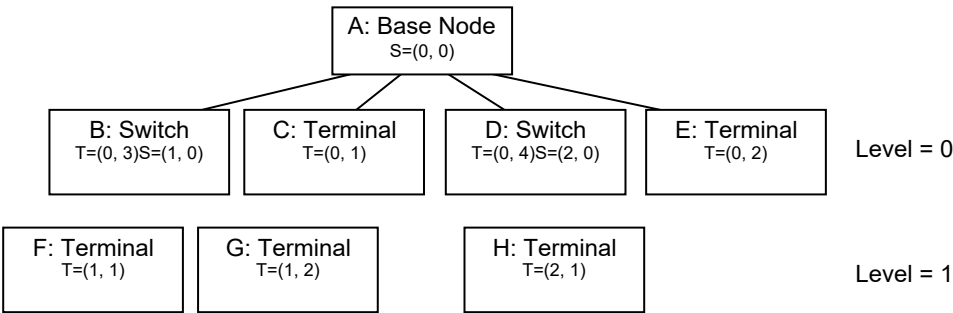


Figure 31 – Example of address resolution: phase 4

4.2.3 Broadcast and multicast addressing

Multicast and broadcast addresses are used for communicating data to multiple Nodes. There are several broadcast and multicast address types, depending on the context associated with the traffic flow. Table 13 describes different broadcast and multicast addressing types and the SID and LNID fields associated with each one.

Table 13 - Broadcast and multicast address

Type	LNID	Description
Broadcast	0x3FFF	Using this address as a destination, the packets should reach every Node of the Subnetwork.
Multicast	0x3FFE	This type of address refers to multicast groups. The multicast group is defined by the LCID.
Unicast	not 0x3FFF not 0x3FFE	The address of this type refers to the only Node of the Subnetwork whose SID and LNID match the address fields.

4.3 MAC functional description

4.3.1 Service Node start-up

At functional level, Service Node starts in Disconnected functional state in a specific initial band (PLC) and/or channel (RF). Disconnected nodes may activate a Band Scanning process on the PLC medium (see 4.3.1.2) and/or a Channel Scanning process on the RF medium (see 4.3.1.3). The activation of a mechanism in one medium is independent from the activation in the other medium. Both activations may be done in parallel. If the Band Scanning process is activated on the PLC medium, the device will try to detect the optimal band (PLC) in which an existing Subnetwork can be joined. The mechanisms adopted to select the initial band (PLC) and to activate the Band Scanning process on the PLC medium are beyond the scope of this specification. If the Channel Scanning process is activated on the RF medium, the device will try to detect the optimal channel

(RF) in which an existing Subnetwork can be joined. The mechanisms adopted to select the initial channel (RF) and to activate the Channel Scanning process on the RF medium are beyond the scope of this specification. It should be observed that the time for network convergence increases with the number of bands (PLC) and/or channels (RF) involved in the scanning processes. So this number should be chosen with care and the bands/channels used for this purpose shall be known in the network.

Each Service Node shall maintain a Switch table that is updated with the reception of a beacon from any new Switch Node. Based on local implementation policies, a Service Node may select any Switch Node from the Switch table and proceed with the Registration process with that Switch Node. The criterion for selecting a Switch Node from the Switch table is beyond the scope of this specification

On the selection of a specific Switch Node, a Service Node shall start a Registration process by transmitting the REG control packet (4.4.2.6.3) to the Base Node. The Switch Node through which the Service Node intends to carry out its communication is indicated in the REG control packet.

When a disconnected node registers to a network, the PLC band scanning and RF channel scanning algorithms specified in 4.3.1.2 and 4.3.1.3, if activated, stop. The registered node, in the medium it is not connected to the network, may decide to adopt a further (PLC band or RF channel) scanning mechanism to possibly improve listening for PNPDU. This mechanism is beyond the scope of this specification.

4.3.1.1 PNPDU transmission

A Service Node shall listen on the initial band (PLC) and/or channel (RF) for at least *macMinSwitchSearchTime* before deciding that no beacon is being received. It may optionally add some random variation to *macMinSwitchSearchTime*, but this variation cannot be more than 10% of *macMinSwitchSearchTime*. If no beacons are received in this time, the Service Node shall broadcast a PNPDU.

PNPDUs are transmitted when a Service Node is not time synchronized to an existing Subnetwork, therefore there are chances that they may collide with contention-free transmissions in a nearby Subnetwork.

In the PLC medium, a Service Node shall therefore necessarily transmit PNPDU in DBPSK_CC modulation scheme before deciding to transmit them in one of the ROBUST modulation schemes (*forcing PHY BC frame format when channel 1 is used*), if such a modulation scheme is implemented in the device. The decision making's algorithm on transitioning from one modulation scheme to other when transmitting PNPDU and scanning for Subnetworks are left to individual implementations.

So as not to flood the network with PNPDU, especially in cases where several devices are powered up at the same time, the Disconnected Nodes shall reduce the PNPDU transmission rate when they receive PNPDU from other sources. Disconnected Nodes shall transmit, for each available medium, at least one PNPDU per *macPromotionMaxTxPeriod* units of time and no more than one PNPDU per *macPromotionMinTxPeriod* units of time (time between PNPDU). The algorithm used to decide the PNPDU transmission rate is left to the implementer.

4.3.1.2 PLC Band Scanning

The functions that may be performed in a *Disconnected* functional state during Band Scanning are: detection of any generic PRIME 1.4 messages or beacons and transmission of PNPDU.

1345 After the waiting time on initial band of *macMinSwitchSearchTime* (described in previous paragraph and
 1346 associated to PNPDU transmission process) the Band scanning process, if activated, starts. If the Service Node
 1347 does not detect traffic in the current band during *macMinBandSearchTime*, the node shall move to the next
 1348 band.

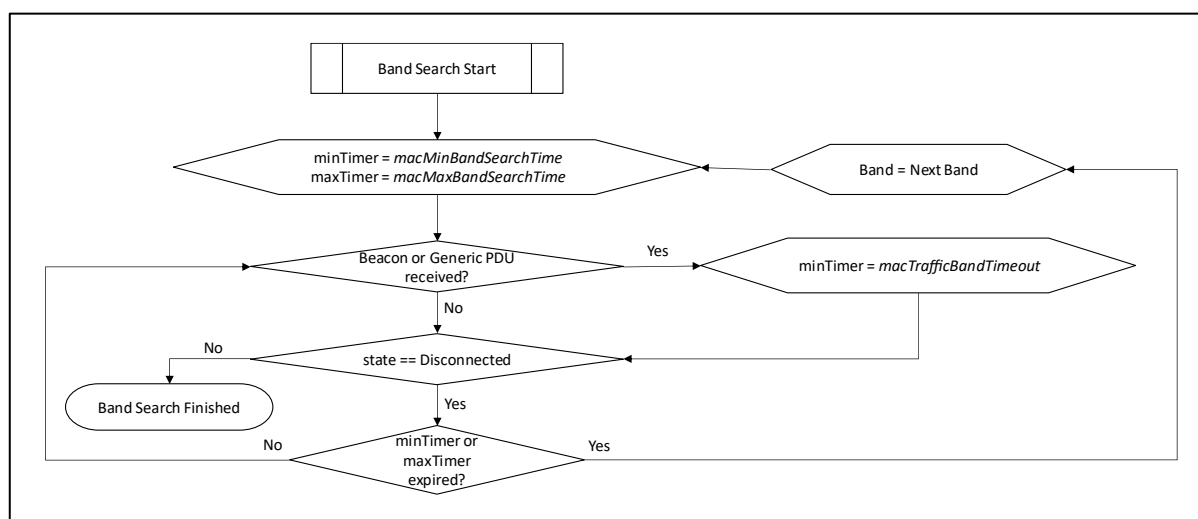
1349 If the node detects traffic in the current band the *macTrafficBandTimeout* is activated (any subsequent
 1350 generic PRIME 1.4 message or beacon detection refreshes the timer), and the moving to next band is
 1351 triggered only when one of the following conditions occurs:

- 1352 - *macTrafficBandTimeout* expiration;
- 1353 or
- 1354 - *macMaxBandSearchTime* expiration.

1355 Whenever a Service Node moves to the next band it restarts the scanning process restarting the timers
 1356 associated to Band Scanning process.

1357 While the node is in *Disconnected* functional state, the band scanning continues going through all the bands
 1358 that are involved in the Band Scanning process. If a registration process is completed with success, the Service
 1359 Node exits from the Band Scanning process. If the node returns in *Disconnected* functional state from *Teminal*
 1360 or *Switch* functional state, the band used during prior registration is considered the new initial band and the
 1361 whole process and associated timers (waiting time on initial band of *macMinSwitchSearchTime* and start of
 1362 Band scanning process) restart.

1363



1364

1365

Figure 32 - Band Scanning algorithm

1366 The optimal values of PIB attributes associated to Band Scanning process depend on several factors to be
 1367 considered at system level, like the Promotion and PNPDU sending processes of other devices present in the
 1368 network. As general recommendations:

- A Service Node, during Band Scanning process, should be able to transmit at least one PNPDU and wait for possible promotion request packets consequently generated by nearby Nodes, if present in the network, before the expiration of *macMinBandSearchTime*;
- Since the minimum frequency of a beacon is the superframe duration, the *macTrafficBandTimeout* should be bigger than this value;
- The *macMaxBandSearchTime* value should be configured in order to permit to Service Node, if beacons are detected in current band, to generate a reasonable number of registration requests (taking into account the MAC Control Packet retransmission settings).

4.3.1.3 RF Channel Scanning

For networks with RF channel hopping, the prescriptions of section 4.6.11.5 hold. The mechanism of Channel Scanning on the RF medium described in this section only applies to networks where RF channel hopping is not used.

The mechanism (see Figure 33) uses the same algorithm described in section 4.3.1.2 for the PLC medium. Hence the same description and recommendations of 4.3.1.2 also apply for the RF medium.

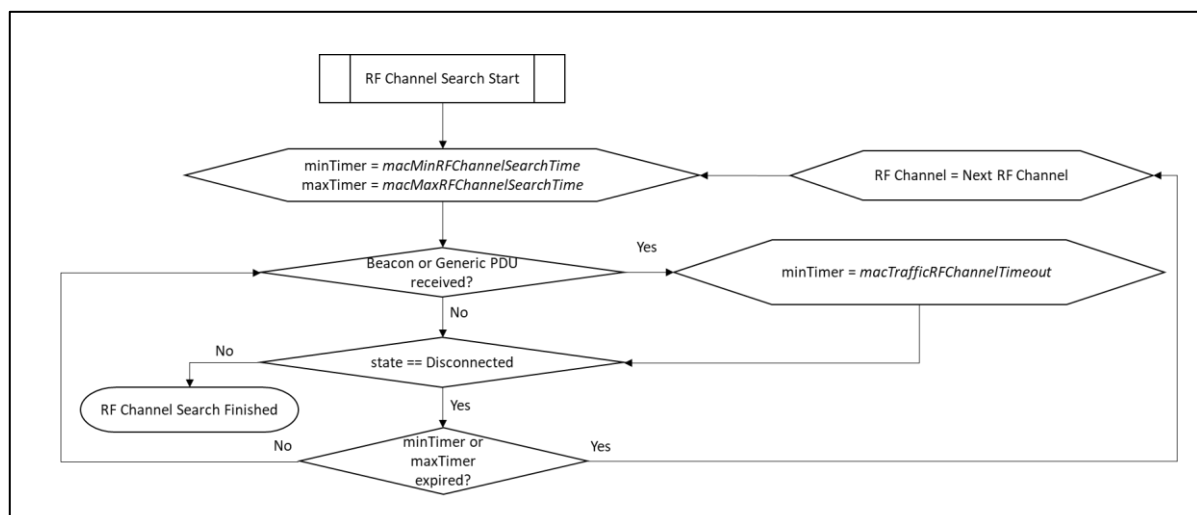


Figure 33 - RF Channel Scanning Algorithm

The RF Channel Scanning attributes *macTrafficRFChannelTimeout*, *macMinRFChannelSearchTime*, *macMaxRFChannelSearchTime* have the same role played by *macTrafficBandTimeout*, *macMinBandSearchTime*, *macMaxBandSearchTime* in PLC Band Scanning and described in section 4.3.1.2.

4.3.2 Starting and maintaining Subnetworks

Base Nodes are primarily responsible for setting up and maintaining a Subnetwork. They would operate in a band comprising of one or more channels. Implementations claiming compliance with this specification shall support at least the mandatory bands required in the respective conformance specification. Base Nodes perform the following functions in order to setup and maintain a Subnetwork:

- **Beacon transmission.** The Base Node and all Switch Nodes in the Subnetwork shall broadcast beacons at fixed intervals of time. The Base Node shall always transmit at least one beacon per super-frame. Switch Nodes shall transmit beacons with a frequency prescribed by the Base Node at the time of their promotion, which would also be at-least one beacon per super-frame.
- **Promotion and demotion of Terminals and switches.** All promotion requests generated by Terminal Nodes upon reception of PNPDUs are directed to the Base Node. The Base Node maintains a table of all the Switch Nodes on the Subnetwork and allocates a unique SID to new incoming requests. Upon reception of multiple promotion requests, the Base Node can, at its own discretion, reject some of the requests. Likewise, the Base Node is responsible for demoting registered Switch Nodes. The demotion may either be initiated by the Base Node (based on an implementation-dependent decision process) or be requested by the Switch Node itself.
- **Registration management.** The Base Node receives Registration requests from all new Nodes trying to be part of the Subnetwork it manages. The Base Node shall process each Registration request it receives and respond with an accept or a reject message. When the Base Node accepts the registration of a Service Node, it shall allocate an unique NID to it to be used for all subsequent communication on the Subnetwork. Likewise, the Base Node is responsible for deregistering any registered Service Node. The unregistration may be initiated by the Base Node (based on an implementation-dependent decision process) or requested by the Service Node itself.
- **Connection setup and management:** The MAC layer specified in this document is connection-oriented, implying that data exchange is necessarily preceded by connection establishment. The Base Node is always required for all connections on the Subnetwork, either as an end point of the connection or as a facilitator (direct connections; Section 4.3.6) of the connection.
- **Channel access arbitration.** The usage of the channel by devices conforming to this specification may be controlled and contention-free at certain times and open and contention-based at others. The Base Node prescribes which usage mechanism shall be in force at what time and for how long. Furthermore, the Base Node shall be responsible for assigning the channel to specific devices during contention-free access periods.
- **Distribution of random sequence for deriving encryption keys.** When using Security Profile 1 (see 4.3.8.1), all control messages in this MAC specification shall be encrypted before transmission. Besides control messages, data transfers may be optionally encrypted as well. The encryption key is derived from a 128-bit random sequence. The Base Node shall periodically generate a new random sequence and distribute it to the entire Subnetwork, thus helping to maintain the Subnetwork security infrastructure.
- **Multicast group management.** The Base Node shall maintain all multicast groups on the Subnetwork. This shall require the processing of all join and leave requests from any of the Service Nodes and the creation of unsolicited join and leave messages from Base Node application requests.

4.3.3 Channel Access

4.3.3.1 MAC Frames

Time is divided into composite units of abstraction for channel usage, called MAC Frames. Composition of a MAC frame is shown in Figure 34. A frame broadly comprises of two parts:

- **Contention Free Part (CFP):** This is the first part of a frame. Only devices that are explicitly granted permission by Base Node are allowed to transmit in CFP. Devices allocated CFP time are also given

start and end time between which they need to complete their transmission and they are not allowed to use the channel for rest of the CFP duration.

- Shared Contention Period (SCP): This is the second half of a frame following the CFP where devices are free to access the channel, provided they:
 - Comply with CSMA CA algorithm enumerated in section 4.3.3.3.2 (and 4.3.3.3.3 if PHY FSK SUN profile is supported) before transmitting their data
 - Respect SCP boundaries within a MAC Frame, together with the corresponding guard-times.

A guard-time of *macGuardTime* needs to be respected at both, beginning and end of CFP. Note that the length of CFP communicated in a beacon is inclusive of its respective guard-times.

In order to facilitate changes to SCP and CFP times in large networks where beacons may not be transmit in every frame, a notion of super-frame is defined. A super-frame is comprised of *MACSuperFrameLength* number of frames. Each frame is numbered in modulo- *MACSuperFrameLength* manner so as to propagate information of super-frame boundary to every device in the subnetwork.

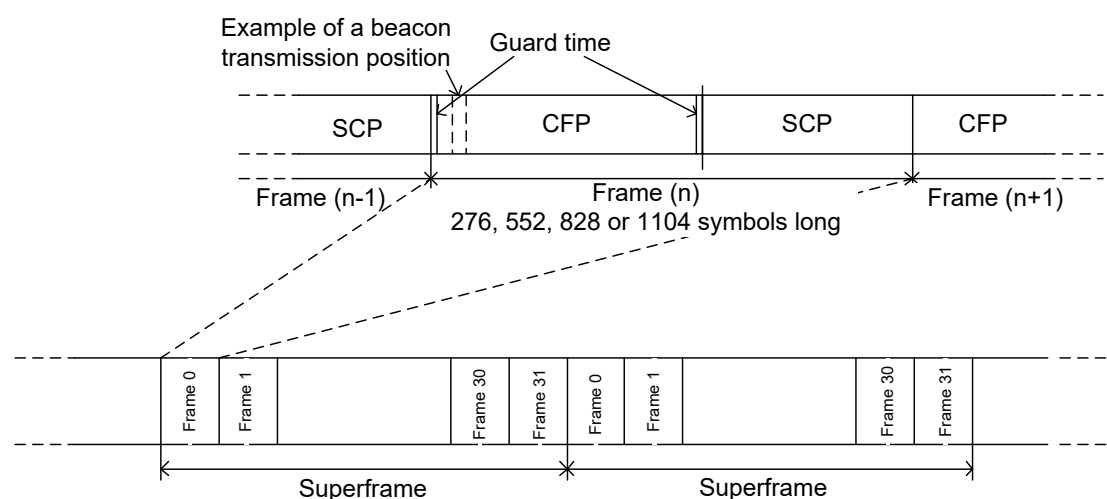


Figure 34 – Structure of a MAC Frame

The length of a frame, *macFrameLength*, together with those of SCP and of CFP are all variable and are defined by Base Node depending on factors such as channel conditions, network size etc. The following mandatory guidelines shall be followed by Base Node implementations while defining the duration of these parameters:

- Frame length can only be one of the four values specified for PIB attribute *macFrameLength*.
- CFP duration within a frame shall at all times be at least ($MACBeaconLength1 + 2 \times macGuardTime$)
- SCP duration within a frame shall at no point in time be less than *MACMinSCPLength*.

Service Nodes may continue to access the channel based on frame organization communicated by Base Node in last received BPDU. Such use of channel also applies to frames when no BPDU is received by the Service Node. Non-reception of BPDU can happen either in normal course when the corresponding Switch Node does not transmit BPDU in every frame or transient channel disturbance resulting in erroneous BPDU reception.

4.3.3.2 Contention-Free Period

Each MAC frame shall have a contention-free period whose duration, in the least, allows transmission of one BPDU.

1465 CFP durations are allocated to Service Nodes in either of the two scenarios:

- 1466 • As part of promotion procedure carried out for a Terminal node. In all such cases, the CFP allocation
1467 will be for usage as beacon-slot by the Service Node being promoted.
1468
- 1469 • As part of CFP allocation process that could be initiated either from Base Node or Service Node, for
1470 use to transport application data.

1471 Service Nodes make channel allocation request in a CFP MAC control packet. The Base Node acts on this
1472 request and responds with a request acceptance or denial. In the case of request acceptance, the Base Node
1473 shall respond with the location of allocation time within MAC frame, the length of allocation time and number
1474 of future MAC frames from which the allocation pattern will take effect. The allocation pattern remains
1475 effective unless there is an unsolicited location change of the allocation period from the Base Node (as a
1476 result of channel allocation pattern reorganization) or the requesting Service Node sends an explicit de-
1477 allocation request using a CFP MAC control packet.

1478 Changes resulting from action taken on a CFP MAC control message that impact overall MAC frame structure
1479 are broadcast to all devices using an FRA MAC control message.

1480 All CFP_ALC_REQ_S requests coming from Terminal or Switch Nodes are addressed to the Base Node.
1481 Intermediate Switch Nodes along the transmission path merely act on the allocation decision by the Base
1482 Node.

1483 Base Nodes may allocate overlapping times to multiple requesting Service Nodes. Such allocations may lead
1484 to potential interference. Thus, a Base Node should make such allocations only when devices that are
1485 allocated channel access for concurrent usage are sufficiently separated. In a multi-level Subnetwork, when
1486 a Service Node that is not directly connected to the Base Node makes a request for CFP, the Base Node shall
1487 allocate CFPs to all intermediate Switch Nodes such that the entire transit path from the source Service Node
1488 to Base Node has contention-free time-slots reserved. The Base Node shall transmit multiple CFP control
1489 packets. The first of these CFP_ALC_IND will be for the requesting Service Node. Each of the rest will be
1490 addressed to an intermediate Switch Node.

1491 **4.3.3.2.1 Beacons**

1492 **4.3.3.2.1.1 General**

1493 Base Node and every other Switch Node in a Subnetwork transmit a Beacon PDU (BPDU) at least once per
1494 super-frame. A BPDU contains administrative and operational information of its respective Subnetwork. Its
1495 contents are enumerated in 4.4.4. Every Service Node in a Subnetwork is required to track beacons as
1496 explained in 4.3.4.1. In addition to using the administrative and operational information, Service Nodes also
1497 synchronize their notion of time based on time of reception of BPDUs.

1498 Since BPDUs are important to keep a Subnetwork running, Base Node and every Switch Node transmitting a
1499 BPDU shall do so using a robust modulation scheme, which is either DBPSK_CC, DBPSK_R or DQPSK_R.
1500 Beacons in DBPSK_CC are transmitted using PHY Frame Type A. Beacons in DBPSK_R and DQPSK_R are
1501 transmitted in PHY Frame Type B. Note that the chosen modulation scheme shall be compliant with the
1502 definition of *macRobustnessManagement*. Every device, including the Base Node shall transmit BPDU with
1503 maximum output power.

4.3.3.2.1.2 Beacon-slots

Unit of time dedicated for transmission of a BPDU is called a beacon-slot. It has a length that is the sum of the beacon length and the *macGuardTime*; where the beacon length is obtained by rounding the effective beacon length to the nearest greatest integer number of OFDM symbols. In the PLC medium, the beacon length depends on the transmission scheme/type and on the number of channels N_{CH} that are concurrently used. In the RF medium, the beacon length depends on the encoding scheme, the RF band, the Operating Mode and the presence or absence of the channel hopping feature. Some examples of beacon lengths can be found in Annex D. Note that a beacon slot includes not only the time required to transmit the BPDU but also the *macGuardTime* that is required to ensure minimal separation between successive transmissions.

Note that a Base Node:

- May decide to not use its beacon-slot in every frame implying that it transmits BPDU at less than once per frame frequency
- Will necessarily use its beacon-slot at least once per *MACSuperFrameLength*
- May allocate its beacon-slot location to other switches in its network for frames where it decides to not transmit its BPDU.

Every Switch Node in the Subnetwork needs to have a beacon-slot allocated in order for it to transmit its BPDU. Switch Nodes are allocated a beacon-slot at time of their promotion by the Base Node.

- Beacon-slot allocations shall necessarily be contained within the CFP duration of a frame.
- Base Node shall not make an allocation that causes a Switch to transmit a beacon overlapping in time with beacons generated by its parent node.
- Base Node may time-multiplex beacon-slots i.e. allocate same duration of time to different switches in different frames.
- A Switch Node may request to change the duration of its beacon-slot when it decides to change the modulation scheme of its BPDU.

With the Registration of each new Switch on the Subnetwork, the Base Node may change the modulation, beacon-slot or BPDU transmission frequency (or both) of already registered Switch devices. When such a change occurs, the Base Node transmits an unsolicited PRO_REQ to each individual Switch device that is affected. The Switch device addressed in the PRO_REQ shall transmit an acknowledgement, PRO_ACK, back to the Base Node. During the reorganization of beacon-slots, if there is a change in CFP duration, the Base Node shall transmit an FRA control packet to the entire Subnetwork. The BN also sends a FRA control packet in advance of a change in length of a frame.

Switch devices that receive an FRA control packet shall relay it to their entire control domain because FRA packets are broadcast information about changes to frame structures.

This is required for the entire Subnetwork to have a common understanding of frame structure, especially in regions where the controlling Switch devices transmit BPDUs at frequencies below once per frame.

4.3.3.2.2 Additional considerations for RF PHY profile

Implementers should take care that, in all messages where CSMA-CA does not apply (for instance for transmissions inside the CFP), other MAC strategies may be needed.

4.3.3.3 Shared-contention period

4.3.3.3.1 General

Shared-contention period (SCP) is the time when any device in Subnetwork can transmit data. SCP follows the CFP duration within a frame and its duration is defined by Base Node. Collisions resulting from simultaneous attempt to access the channel are avoided by the CSMA-CA mechanism specified in this section. SCP durations are highlighted by the following key specifications:

- SCP duration within a frame shall at no point in time be less than *MACMinSCPLength*.
- Maximum possible duration of SCP shall be $(\text{macFrameLength} - (\text{MACBeaconLength1} + 2 \times \text{macGuardTime}))$. This is the case of a subnetwork that does not have dedicated CFP requests from any Service Node.

4.3.3.3.2 PLC CSMA-CA algorithm

The CSMA-CA algorithm implemented in devices works as shown in Figure 35.

Implementations start with a random backoff time (*macSCPRBO*) based on the priority of data queued for transmission. *MACPriorityLevels* levels of priority need to be defined in each implementation, with a lower value indicating higher priority. In the case of data aggregation, the priority of aggregate bulk is governed by the highest priority data it contains. The *macSCPRBO* for a transmission attempt is given as below:

$$\text{macSCPRBO} = \text{random} (0, \text{MIN} ((2^{(\text{Priority} + \text{txAttempts} + \text{macCSMAR1})} + \text{macCSMAR2}), (\text{macSCPLength}/2)))$$

or when Robust Modes are supported:

$$\text{macSCPRBO} = \text{random} (0, \text{MIN} ((2^{(\text{Priority} + \text{txAttempts} + \text{macCSMAR1Robust})} + \text{macCSMAR2Robust}), (\text{macSCPLength}/2)))$$

macCSMAR1/macCSMAR1Robust and *macCSMAR2/macCSMAR2Robust* control the initial contention window size. *macCSMAR1/macCSMAR1Robust* helps to increase the contention window size exponentially while *macCSMAR2/macCSMAR2Robust* helps to increase the contention window linearly. A higher value of *macCSMAR1/macCSMAR1Robust* and/or *macCSMAR2/macCSMAR2Robust* is recommended for large networks. It is recommended to not decrease the default values.

Before a backoff period starts, a device should ensure that the remaining SCP time is long enough to accommodate the backoff, the number of iterations for channel-sensing (based on data priority) and the subsequent data transmission. If this is not the case, backoff should be aborted till the SCP starts in the next frame. Aborted backoffs that start in a subsequent frame should not carry *macSCPRBO* values of earlier attempts. *macSCPRBO* values should be regenerated on the resumption of the transmission attempt in the SCP time of the next frame.

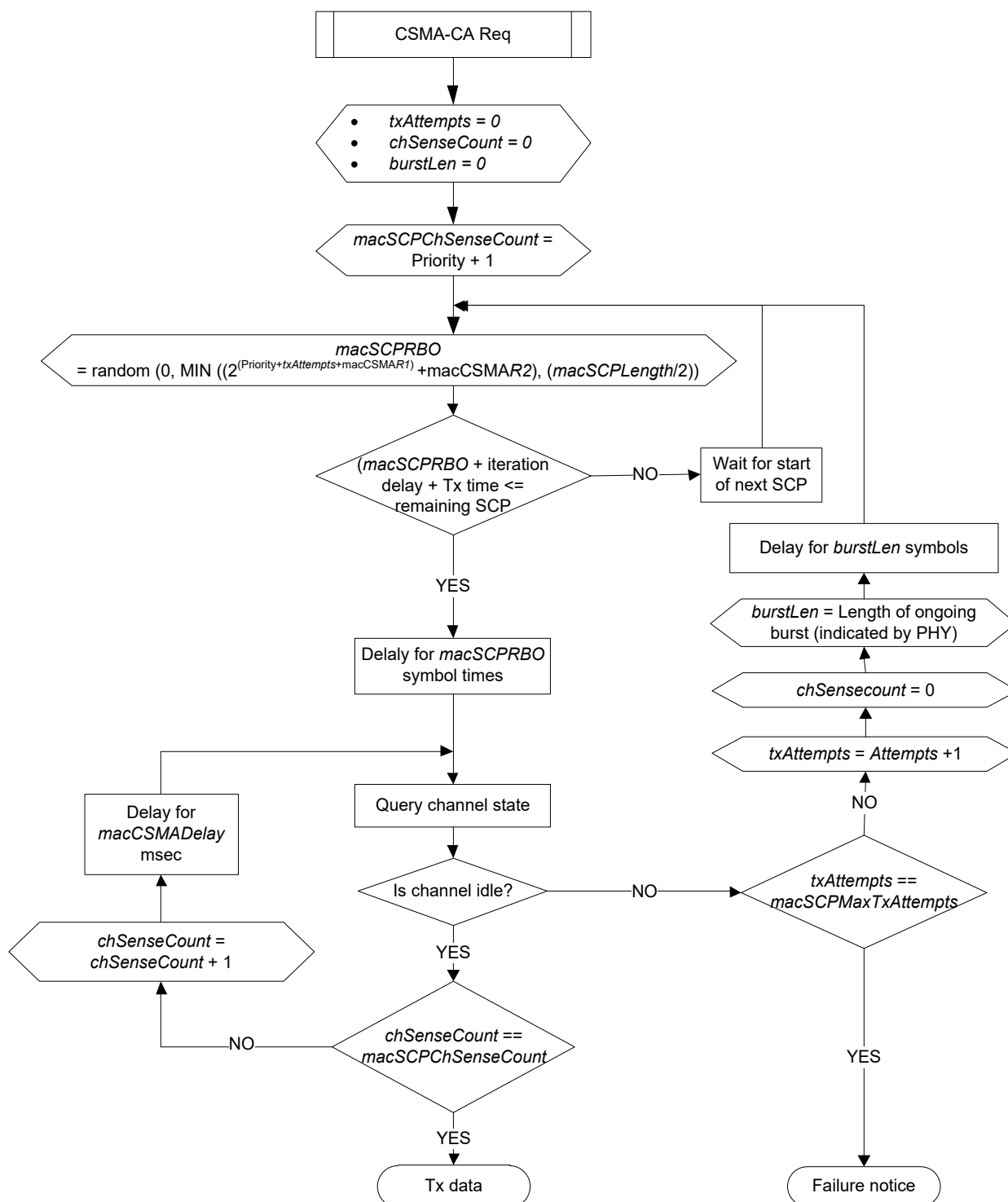


Figure 35 - Flow chart for PL CSMA-CA algorithm

On the completion of *macSCPRBO* symbol time, implementations perform channel-sensing. Channel sensing shall be performed one or more times depending on priority of data to be transmitted. The number of times for which an implementation has to perform channel-sensing (*macSCPChSenseCount*) is defined by the priority of the data to be transmitted with the following relation:

$$macSCPChSenseCount = Priority + 1$$

and each channel sense should be separated by a *macCSMADelay* ms delay.

Note: *macSCPRBO* and *macCSMADelay* follow a different range and different default value depending on the modulation scheme that is intended to be used for a transmission burst. If a device intends to use robust mode for some bursts, the values are conservative to account for extended PHY Frame (Type B) timings. The applicable values are listed in 6.2.3.2. Implementations shall conform to listed range and default value corresponding to the modulation scheme used.

When a channel is sensed to be idle on all *macSCPChSenseCount* occasions, an implementation may conclude that the channel status is idle and carry out its transmission immediately.

During any of the *macSCPChSenseCount* channel-sensing iterations, if the channel is sensed to be occupied, implementations should reset all working variables. The local counter tracking the number of times a channel is found to be busy should be incremented by one and the CSMA-CA process should restart by generating a new *macSCPRBO*. The remaining steps, starting with the backoff, should follow as above.

If the CSMA-CA algorithm restarts *macSCPMaxTxAttempts* number of times due to ongoing transmissions from other devices on the channel, the transmission shall abort by informing the upper layers of CSMA-CA failure.

4.3.3.3.3 CSMA-CA algorithm for SUN FSK PHY profile

When PRIME SUN FSK PHY profile is supported and SUN FSK PPDU are transmitted on the SCP, the slotted version of the CSMA-CA shown in Figure 6-5 of [28] shall be used (see Figure 36). The generic description of the algorithm can be derived from section 6.2.5.1 of [28] with the following assumptions:

- SCP substitutes the concept of CAP and the start of the first backoff period of each device is aligned with the start of the SCP;
- TSCH (Timeslotted Channel Hopping) is disabled;
- CW_0 is equal to 2 as in [28];
- The Battery Life Extension is disabled;
- Writable PIBs *macMinBe*, *macMaxBe*, *macMaxCsmBackoffs* described in Table 106 of this specification shall be implemented;
- MAC constant *aUnitBackOffPeriod* uses the definition reported in this specification in Annex D;
- the statements reported in section 6.2.5.1 of [28], which involve specific IEEE MAC concepts as PAN, Imm-Ack frames, Enh-Ack frames, Data frames, acknowledgment of a Data Request command, are ignored.

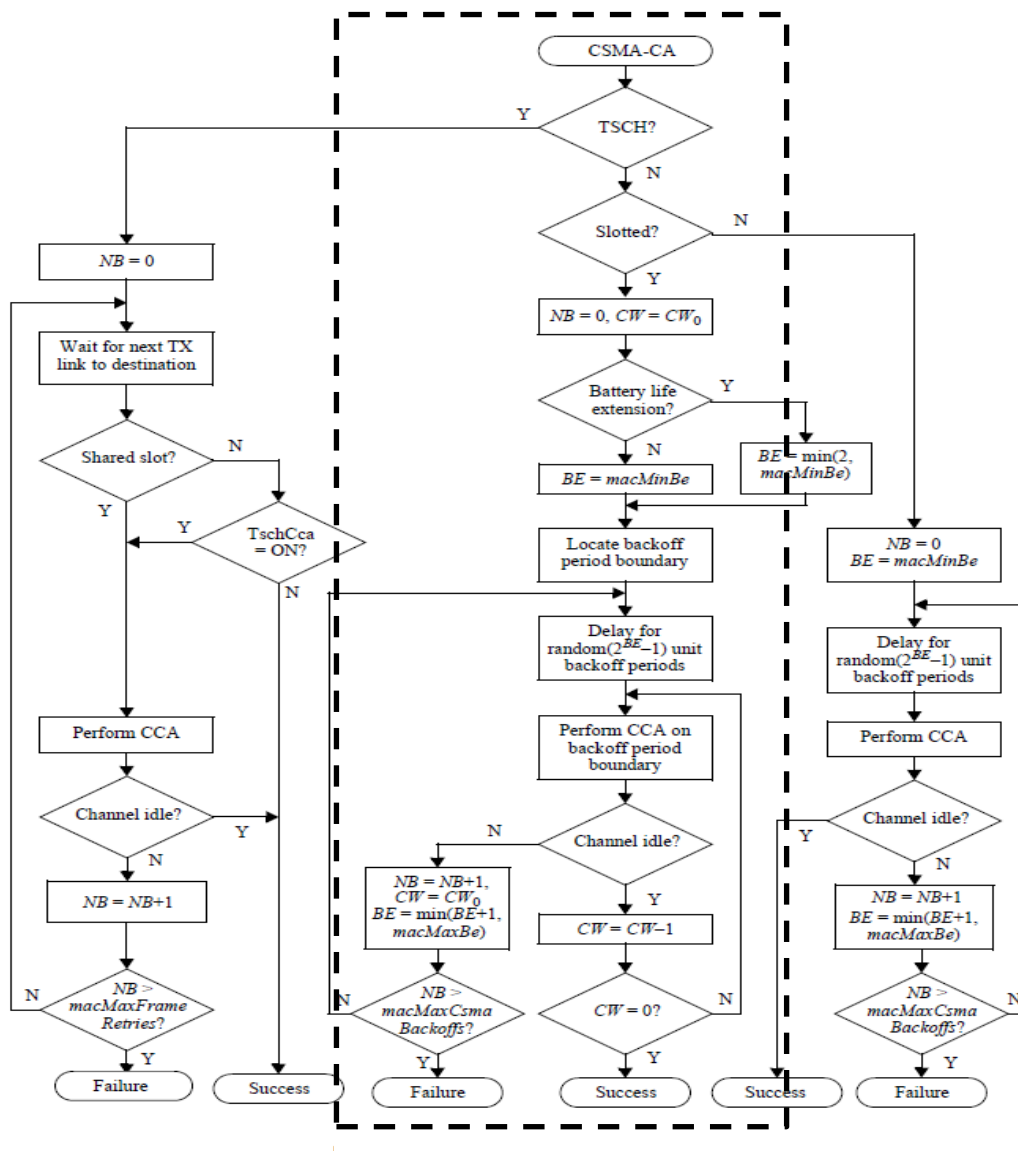


Figure 36 - Flow chart for SUN FSK PHY CSMA-CA algorithm

4.3.3.3.4 MAC control packet transmission

MAC control packets (4.4.2.6) shall follow the following channel access rules:

- Always transmit in SCP
- Use priority level of *MACCtrlPktPriority*
- The MAC Control Packets shall be transmitted in a modulation scheme robust enough to reach the receiving peer but no less robust than *MACCtrlPktMinRobustness*.
- Transmitted with PHY Frame Type B for DBPSK_R and DQPSK_R. For all other modulation schemes, control packets are transmitted using PHY Frame Type A

1630 4.3.4 Tracking switches and peers

1631 4.3.4.1 Tracking switches

1632 Service Nodes shall keep track of all neighboring Switch Nodes by maintaining a list of beacons received. Such
1633 tracking shall keep a Service Node updated on reception signal quality from Switch Nodes other than the one
1634 to which it is connected, thus making it possible to change connection points (Switch Node) to the
1635 Subnetwork if link quality to the existing point of connectivity degrades beyond an acceptable level.

1636 Note that such a change of point of connectivity may be complex for Switch Nodes because of devices
1637 connected through them. However, at certain times, network dynamics may justify a complex reorganization
1638 rather than continue with existing limiting conditions.

1639 4.3.4.2 Tracking disconnected Nodes

1640 Terminals shall process all received PNPDU. When a Service Node is Disconnected, it doesn't have
1641 information on current MAC frame structure so the PNPDU may not necessarily arrive during the SCP. Thus,
1642 Terminals shall also keep track of PNPDU during the CFP or beacon-slots.

1643 On processing a received PNPDU, a Terminal Node may decide to ignore it and not generate any
1644 corresponding promotion request (PRO_REQ_S). Receiving multiple PNPDU can indicate that there is no
1645 other device in the vicinity of Disconnected Nodes, implying that there will be no possibility of new devices
1646 for connecting to the Subnetwork if the Terminal Node does not request promotion itself. A Terminal Node
1647 shall ignore no more than *MACMaxPRNIgnore* PNPDU. After this maximum number of ignored PNPDU the
1648 Terminal Node shall start a Promotion procedure as described in 4.6.3. The time in which the procedure will
1649 start shall be randomly selected in the range of $[0, MACMaxPRNIgnore * macPromotionMinTxPeriod]$ seconds.

1650 4.3.4.3 Tracking switches under one node

1651 Service Nodes in *Switch* functional state shall keep track of the Switches under their tree by maintaining the
1652 *macListSwitchTable*. Maintaining this information is sufficient for switching because traffic to/from Terminal
1653 Nodes will also contain the identity of their respective Switch Nodes (PKT.SID). Thus, the switching function
1654 is simplified in that maintaining an exhaustive listing of all Terminal Nodes connected through it is not
1655 necessary. After promotion Switch Nodes start with no entries in their switching table.

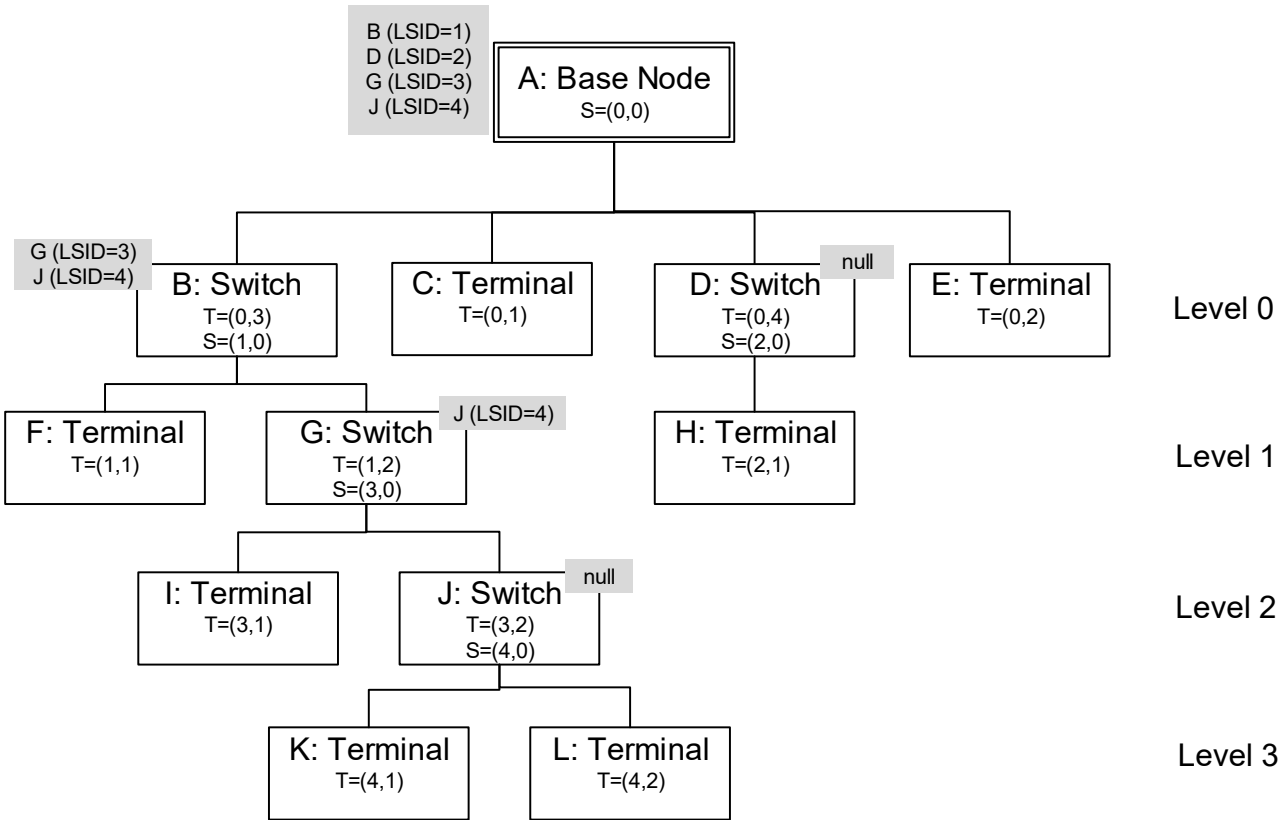


Figure 37 –Switching tables examples

One Switch Node shall include in the switching list the SID of every promoted Terminal node which is directly connected to it or to one Switch already included in the macListSwitchTable. In this case the Node shall create an entry with the stblEntryLSID value equals to the NSID field of the PRO_ACK packet, the stblEntrySID value equal to PKT.SID of the PRO.ACK Packet Header and stblEntryLNID equal to PKT.LNID of the PRO.ACK Packet Header.

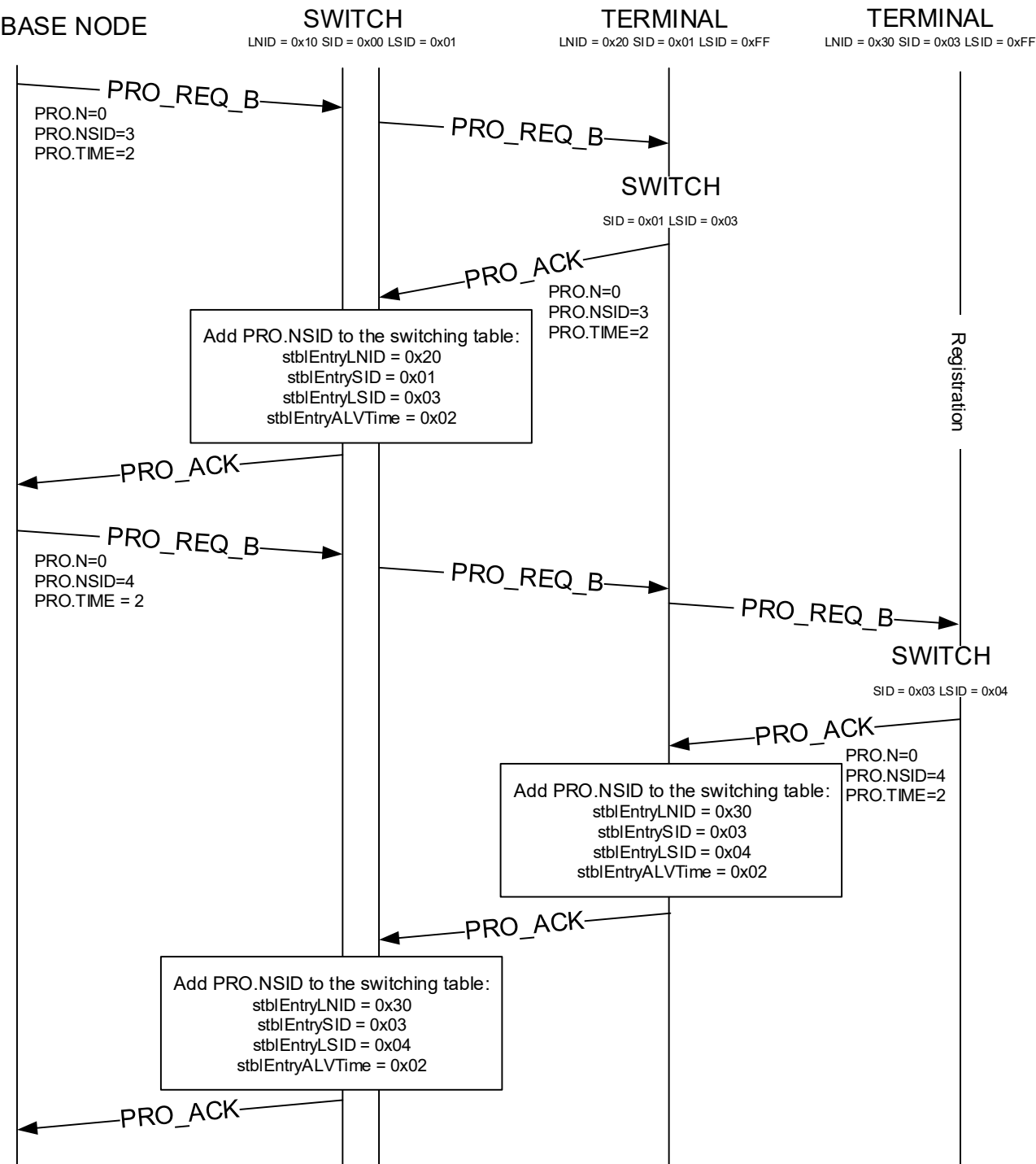


Figure 38 – Filling example for the switching table for Switch of Level 0.

Similarly the Switch Node shall mark the entry to be removed from the list the SID when a node is removed or unregistered. This is be done by listening the PRO_DEM_B, PRO_DEM_S, REG_UNR_B or REG_UNR_S packets.

Each entry of the *macListSwitchTable* also contains the information related to the Alive time related to the Switch node. The stblEntryALVTime is updated with the TIME field received during the promotion, beacon robustness change or the keep alive procedures. The Switch Node shall also maintain the T_{keep-alive} timer for

every Switch under its tree. The Switch Node shall refresh the timers as specified in section 4.6.5. If the $T_{keep-alive}$ timer expires the entry in the *macListSwitchTable* shall be marked to be removed.

Every time an entry is marked to be removed, the Switch Node shall check if the *stblEntrySID* of other entries is equal to the *stblEntryLSID*. In these cases all the entries shall be marked to be removed, meaning that one entire branch has left the network.

When one entry is marked to be removed the Switch Node shall wait (*macCtrlMsgFailTime* + *macMinCtrlReTxTimer*) seconds. This time ensures that all retransmit packets which use the SID have left the Subnetwork. When the timer expires the table entry shall be removed.

4.3.5 Switching

4.3.5.1 General

On a Subnetwork, the Base Node cannot communicate with every Node directly. Switch Nodes relay traffic to/from the Base Node so that every Node on the Subnetwork is effectively able to communicate with the Base Node. Switch Nodes selectively forward traffic that originates from or is destined to one of the Service Nodes in its control hierarchy. All other traffic is discarded by Switches, thus optimizing traffic flow on the network.

Different names of MAC header and packets are used in this section. Please refer to the section 4.4.2 to find their complete specification.

4.3.5.2 Switching process

Switch Nodes forward traffic to their control domain in a selective manner. The received data shall fulfill the conditions listed below for it to be switched. If the conditions are not met, the data shall be silently discarded.

Downlink packets (HDR.DO=1) shall meet any of the following conditions in order to be switched:

- Destination Node of the packet is connected to the Subnetwork through this Switch Node, i.e. PKT.SID is equal to this Switch Node's SID or its switching table contains an entry for PKT.SID.
- The packet has broadcast destination (PKT.LNID = 0x3FFF) and was sent by the Switch this Node is registered through (PKT.SID=SID of this Switch Node).
- The packet has a multicast destination (PKT.LNID=0x3FFE), it was sent by the Switch this Node is registered through (PKT.SID=SID of this Switch Node) and at least one of the Service Nodes connected to the Subnetwork through this Switch Node is a member of the said multicast group, i.e. LCID specifies a group that is requested by any downstream Node in its hierarchy.

Uplink packets (HDR.DO=0) shall meet either of the following conditions in order to be switched:

- The packet source Node is connected to the Subnetwork through this Switch Node, i.e. PKT.SID is equal to this Switch Node's SID or its switching table contains an entry for PKT.SID.
- The packet has a broadcast or multicast destination (PKT.LNID = 0x3FFF or 0x3FFE) and was transmitted by a Node connected to the Subnetwork through this Switch Node i.e. PKT.SID is equal to this Switch Node's SID or its switching table contains an entry for PKT.SID.

If a packet meets previous conditions, it shall be switched. For unicast packets, the only operation to be performed during switching is to queue it to be resent in a MAC PDU with the same HDR.DO.

1708 In case of broadcast or multicast packets, the PKT.SID must be replaced with the Switch Node's LSID for
1709 Downlink packets. Uplink packets shall be resent unchanged.

1710 When switching packets, the Switch Node shall ignore the length and the values of the reserved bits in the
1711 received packet. The destination Node is responsible for checking that the length and the reserved bits match
1712 the expected values.

1713 **4.3.5.3 Switching of broadcast packets**

1714 The switching of broadcast MAC frames operates in a different manner to the switching of unicast MAC
1715 frames. Broadcast MAC frames are identified by PKT.LNID=0x3FFF.

1716 When HDR.DO=0, i.e. the packet is an uplink packet, it is unicast to the Base Node. A Switch which receives
1717 such a packet shall apply the scope rules to ensure that it comes from a lower level and, if so, Switch it
1718 upwards towards the base. The rules given in section 4.3.5.2 must be applied. The same modulation scheme
1719 and output power level as used for unicast uplink switching shall be used.

1720 When HDR.DO=1, i.e. the packet is a Downlink packet, it is broadcast to the next level. A Switch which
1721 receives such a packet shall apply the scope rules to ensure that it comes from the higher level and, if so,
1722 switch it further to its Subnetwork. The same modulation scheme as used for beacon transmission at the
1723 maximum output power level implemented in the device shall be used so that all the devices directly
1724 connected to the Switch Node can receive the packet. The rules given in section 4.3.5.2 must be applied. The
1725 Service Node shall also pass the packet up to its MAC SAP to applications which have registered to receive
1726 broadcast packets using the MAC_JOIN service.

1727 When the Base Node receives a broadcast packet with HDR.DO=0, it shall pass the packet up its MAC SAP to
1728 applications which have registered to receive broadcast packets. The Base Node shall also transmit the packet
1729 as a Downlink packet, i.e. HDR.DO=1, using the same modulation scheme as used for beacon transmission at
1730 the maximum output power level and following the rules given in section 4.3.5.2.

1731 **4.3.5.4 Switching of multicast packets**

1732 Switch Nodes shall maintain a multicast switching table. This table contains a list of multicast group LCIDs
1733 that have members connected to the Subnetwork through the Switch Node. The LCID of multicast traffic in
1734 both Downlink and uplink directions is checked for a matching entry in the multicast switching table.
1735 Multicast traffic is only switched if an entry corresponding to the LCID is available in the table; otherwise, the
1736 traffic is silently discarded.

1737 A multicast switching table is established and managed by examining the multicast join messages (MUL
1738 control packet) which pass through the Switch. On a successful group join from a Service Node in its control
1739 hierarchy, a Switch Node adds a new multicast Switch entry for the group LCID, where necessary. An entry
1740 from the multicast switching table can be removed by the Base Node using the multicast leave procedure
1741 (see section 4.6.7.4.2). All entries from the multicast switching table shall be removed when a switch is
1742 demoted or unregistered. The multicast packet switching process depends on the packet direction.

1743 When HDR.DO=0 and PKT.LNID=0x3FFE, i.e. the packet is an uplink multicast packet, it is unicast towards the
1744 Base Node. A Switch Node that receives such a packet shall apply the scope rules to ensure it comes from a
1745 lower hierarchical level and, if so, switch it upwards towards the Base Node. No LCID-based filtering is

1746 performed. All multicast packets are switched, regardless of any multicast Switch entries for the LCID. The
1747 rules given in section 4.3.5.2 must be applied. The same modulation scheme and output power level as used
1748 for unicast uplink switching shall be used.

1749 When HDR.DO=1 and PKT.LNID=0x3FFE, i.e. the packet is a Downlink multicast packet, the multicast
1750 switching table is used. If there is an entry with the LCID corresponding to PKT.LCID in the packet, the packet
1751 is switched downwards to the part of Subnetwork controlled by this switch. The multicast traffic shall be
1752 relayed using a modulation scheme which is robust enough to ensure that all direct children which are part
1753 of the multicast group or which need to switch the multicast traffic can receive the packet. As a guideline,
1754 the same modulation scheme as used for beacon transmission at the maximum output power level can be
1755 used. The rules given in section 4.3.5.2 shall be applied. If the Service Node is also a member of the multicast
1756 group, it shall also pass the packet up its MAC SAP to applications which have registered to receive the
1757 multicast packets for that group.

1758 When the Base Node receives a multicast packet with HDR.DO=0 and it is a member of the multicast group,
1759 it shall pass the packet up its MAC SAP to applications which have registered to receive multicast packets for
1760 that group. The Base Node shall switch the multicast packet if there is an appropriate entry in its multicast
1761 switching table for the LCID, transmitting the packet as a Downlink packet, i.e. HDR.DO=1. To transmit a
1762 downlink multicast packet by the Base Node the same rules apply as for transmitting a downlink multicast
1763 packet by a switch.

1764 **4.3.6 Direct connections**

1765 **4.3.6.1 Direct connection establishment**

1766 The direct connection establishment is a little different from a normal connection although the same packets
1767 and processes are used. It is different because the initial connection request may not be acknowledged until
1768 it is already acknowledged by the target Node. It is also different because the CON_REQ_B packets shall carry
1769 information for the “direct Switch” to update the “direct switching table”.

1770 A direct switch is not different than a general switch. It is only a logical distinction of identifying the first
1771 common switch between two service-nodes that need to communicate with each other. Note that in absence
1772 of such a common switch, the Base Node would be the direct switch.

1773 There are two different scenarios for using directed connections. These scenarios use the network shown in
1774 Figure 39.

1775 The first is when the source Node does not know the destination Service Node’s EUI-48 address. The Service
1776 Node initiates a connection to the Base Node and the Base Node Convergence layer redirects the connection
1777 to the correct Service Node.

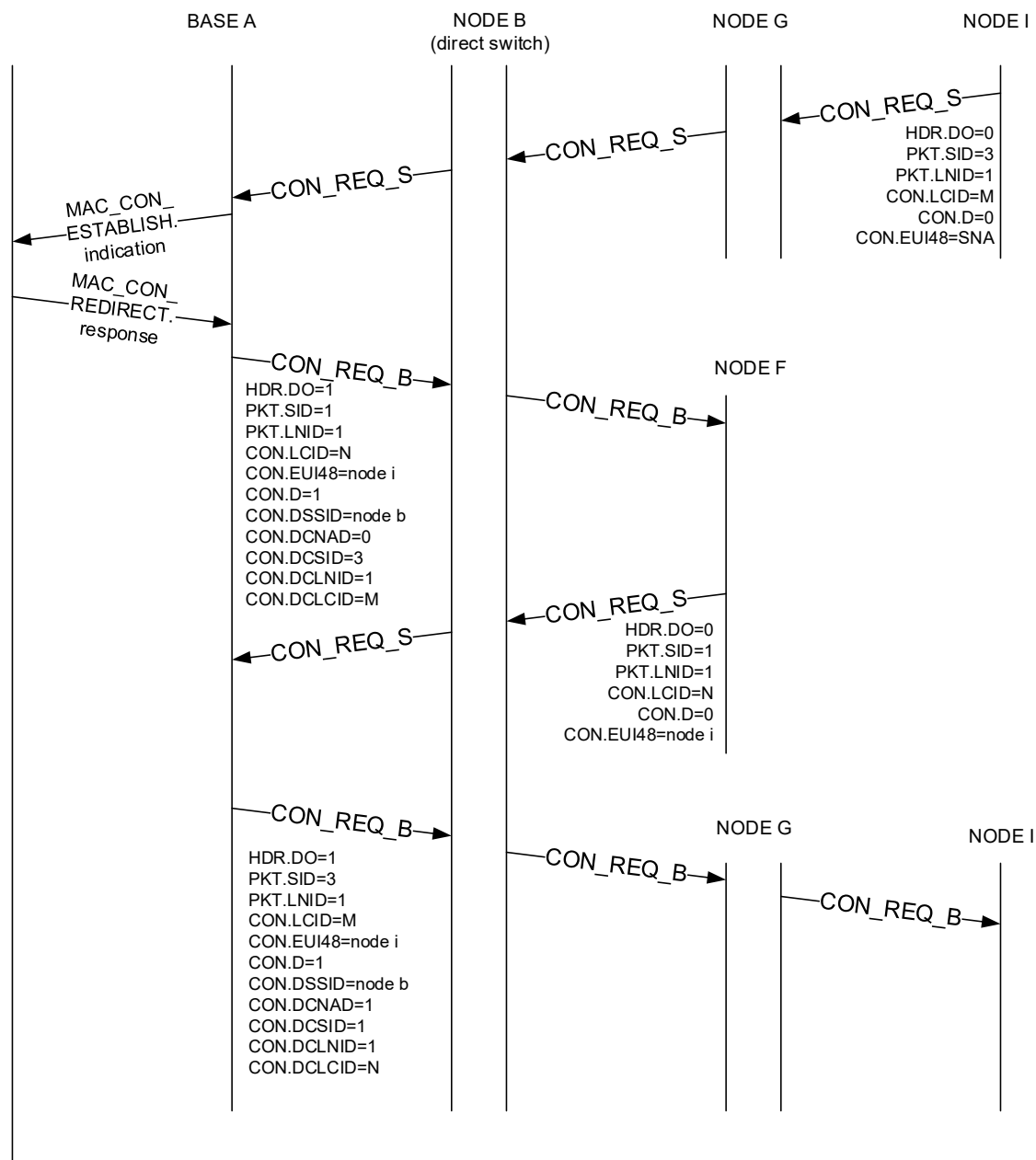


Figure 39 – Directed Connection to an unknown Service Node

1778
1779

1780 The steps to establish a direct connection, as shown in Figure 39, shall be:

- ```

1781 • When Node I tries to establish connection with Node F, it shall send a normal connection request
1782 (CON_REQ_S).
1783 • Then, due to the fact that the Base Node knows that F is the target Service Node, it should send a
1784 connection request to F (CON_REQ_B). This packet will carry information for direct Switch B to include
1785 the connection in its direct switching table.
1786 • F may accept the connection. (CON_REQ_S).
1787 • Now that the connection with F is fully established, the Base Node will accept the connection with I
1788 (CON_REQ_B). This packet will carry information for the direct Switch B to include in its direct
1789 switching table.

```

1790 After finishing this connection-establishment process, the direct Switch (Node B) should contain a direct  
1791 switching table with the entries shown in Table 14.

1792 **Table 14 - Direct connection example: Node B's Direct switching table**

| Uplink |      |      | Downlink |       |       |     |
|--------|------|------|----------|-------|-------|-----|
| SID    | LNID | LCID | DSID     | DLNID | DLCID | NAD |
| 1      | 1    | N    | 3        | 1     | M     | 0   |
| 3      | 1    | M    | 1        | 1     | N     | 1   |

1793

1794 The direct switching table should be updated every time a Switch receives a control packet that meets the  
1795 following requirements.

- 1796 • It is CON\_REQ\_B packet: HDR.DO=1, CON.TYPE=1 and CON.N=0;
- 1797 • It contains “direct” information: CON.D=1;
- 1798 • The direct information is for itself: CON.DSSID is the SID of the Switch itself.

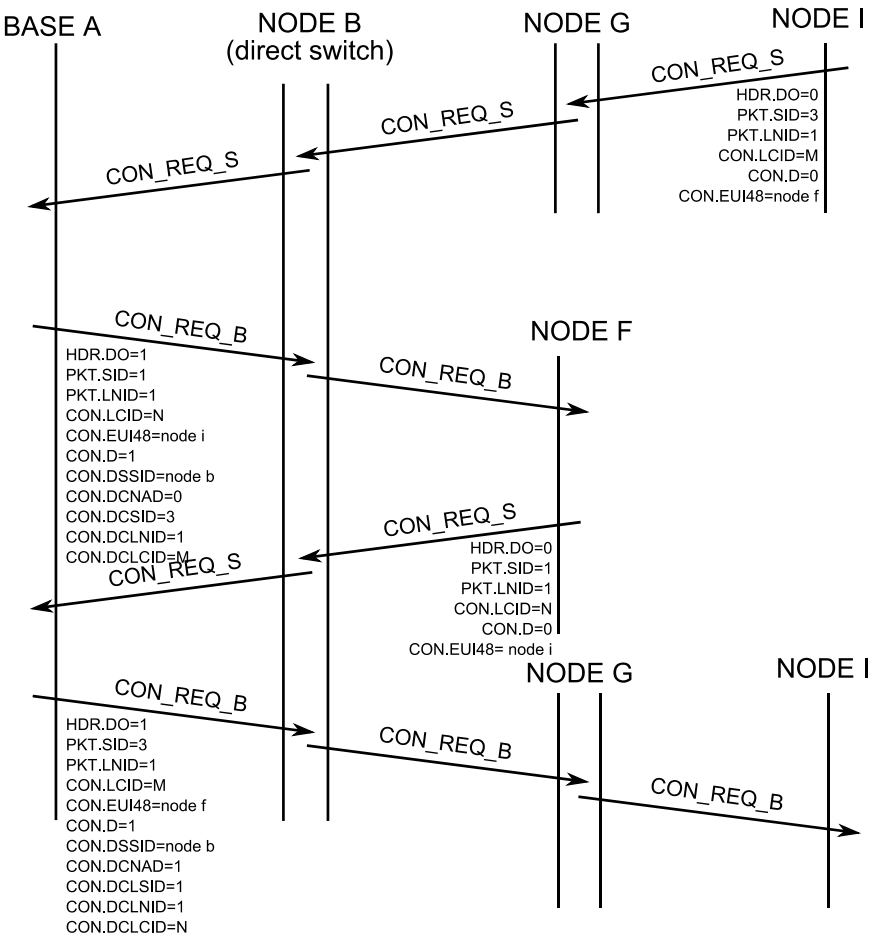
1799 Then, the direct switching table is updated with the information:

- 1800 • Uplink (SID, LNID, LCID) = (PKT.SID, PKT.LNID, CON.LCID);
- 1801 • Downlink (SID, LNID, LCID, NAD) = (CON.DCSID, CON.DCLNID, CON.DCLCID, CON.DCNAD).

1802 The connection closing packets should be used to remove the entries.

1803 The second scenario for using directed connections is when the initiating Service Node already knows the  
1804 destination Service Node's EUI-48 address. In this case, rather than using the Base Node's address, it uses  
1805 the Service Node's address. In this case, the Base Node Convergence layer is not involved. The Base Node  
1806 MAC layer connects Service Node I directly to Service Node F. The resulting Switch table entries are identical  
1807 to the previous example. The exchange of signals is shown in Figure 40.

1808



1809

1810

Figure 40 - Example of direct connection: connection establishment to a known Service Node

1811

#### 4.3.6.2 Direct connection release

1812

The release of a direct connection is shown in Figure 41. The signaling is very similar to connection establishment for a direct connection. The D fields are used to tell the direct Switch which entries it should remove. The direct switching table should be updated every time a Switch receives a control packet that meets the following requirements.

1813

1814

1815

1816

- It is CON\_CLOSE\_B packet: HDR.DO=1, CON.TYPE=1 and CON.N=1;
- It contains "direct" information: CON.D=1;
- The direct information is for itself: CON.DSSID is the SID of the Switch itself.

1817

1818

1819

Then, the direct switching table entry with the following information is removed:

1820

- Uplink (SID, LNID, LCID) = (PKT.SID, PKT.LNID, CON.LCID);
- Downlink (SID, LNID, LCID, NAD) = (CON.DCSID, CON.DCLNID, CON.DCLCID, CON.DCNAD).

1821

1822

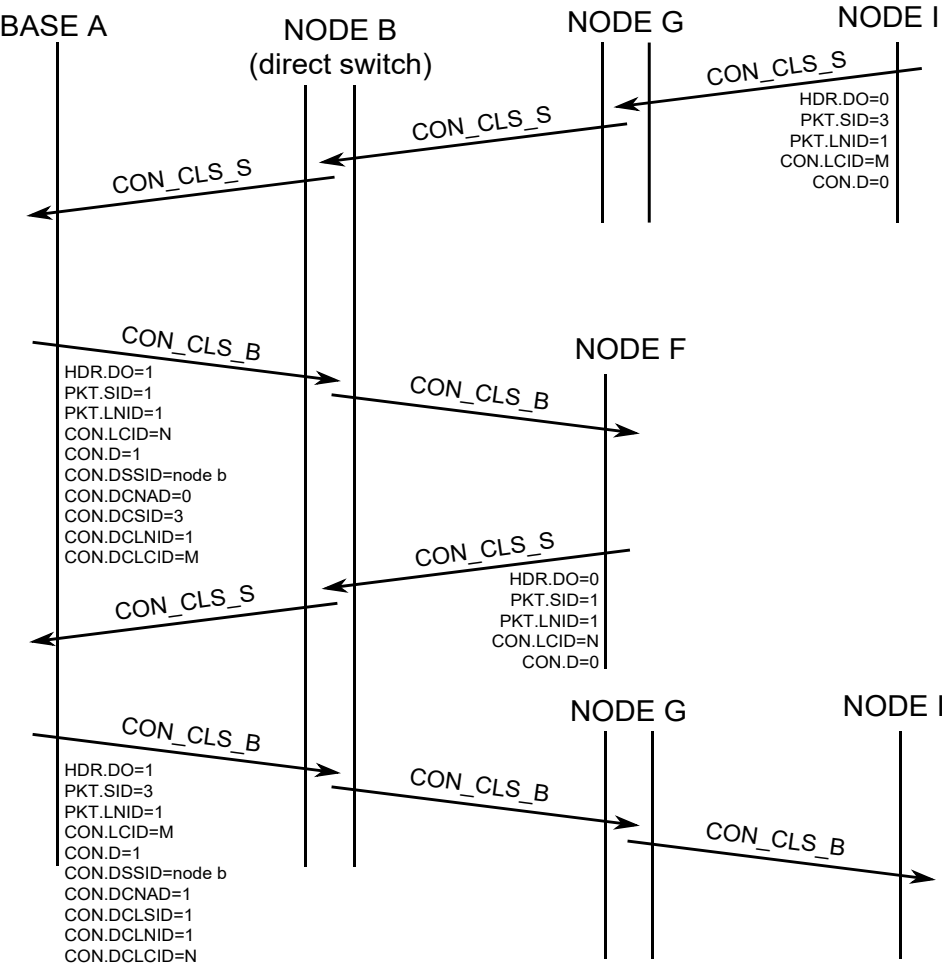


Figure 41 - Release of a direct connection

### 4.3.6.3 Direct connection switching

As explained in section 4.3.5.2, the normal switching mechanism is intended to be used for forwarding communication data between the Base Node and each Service Node. The “direct switching” is a mechanism to let two Nodes communicate with each other, switching the packets in a local way, i.e. without passing through the Base Node. It is not a different form of packet-switching, but rather an additional feature of the general switching process.

The first shared Switch in the paths that go from two Service Nodes to the Base Node will be called the “direct Switch” for the connections between the said Nodes. This is the Switch that will have the possibility of performing the direct switching to make the two Nodes communicate efficiently. As a special case, every Switch is the “direct Switch” between itself and any Node that is lower down in the hierarchy.

The “direct switching table” is a table every Switch should contain in order to perform the direct switching. Each entry on this table is a direct connection that must be switched directly. It is represented by the origin CID and the destination CID of the direct connection. It is not a record of every connection identifier lower down in its hierarchy, but contains only those that should be directly switched by it. The Destination Node’s

1840 ability to receive aggregated packets shall also be included in the “direct switching table” in order to fill the  
1841 PKT.NAD field.

#### 1842 **4.3.6.4 Direct switching operation**

1843 If a Switch receives an uplink (HDR.DO=0) MAC frame that is to be switched (see section 4.3.5.2 for the  
1844 requirements) and its address is in the direct switching table, then the procedure is as follows:

- 1845 • Change the (SID, LNID, LCID, NAD) by the Downlink part of the entry in the direct switching table.
- 1846 • Queue the packet to be transmitted as a Downlink packet (HDR.DO=1).

### 1847 **4.3.7 Packet aggregation**

#### 1848 **4.3.7.1 General**

1849 The GPDU may contain one or more packets. The functionality of including multiple packets in a GPDU is  
1850 called packet aggregation. Packet aggregation is an optional part of this specification and devices do not need  
1851 to implement it for compliance with this specification. It is however suggested that devices should implement  
1852 packet aggregation in order to improve MAC efficiency.

1853 To maintain compatibility between devices that implement packet aggregation and ones that do not, there  
1854 must be a guarantee that no aggregation takes place for packets whose data transit path from/to the Base  
1855 Node crosses (an) intermediate Service Node(s) that do(es) not implement this function. Information about  
1856 the aggregation capability of the data transit path is exchanged during the Registration process (4.6.1). A  
1857 registering Service Node notifies this capability to the Base Node in the REG.CAP\_PA field (1 bit, see Table  
1858 22) of its REG\_REQ message. It gets feedback from the Base Node on the aggregation capability of the whole  
1859 Downlink transit path in the REG.CAP\_PA field of the REG\_RSP message.

1860 Based on initial information exchanged on Registration, each subsequent data packet in either direction  
1861 contains aggregation information in the PKT.NAD field. In the Downlink direction, the Base Node will be  
1862 responsible for filling PKT.NAD based on the value it communicated to the destination Service Node in the  
1863 REG.CAP\_PA field of the REG\_RSP message. Likewise, for uplink data, the source Service Node will fill  
1864 PKT.NAD based on the REG.CAP\_PA field received in the initial REG\_RSP from the Base Node. The last Switch  
1865 shall use the PKT.NAD field to avoid packet aggregation when forwarding the packet to destination Service  
1866 Nodes without packet aggregation capability. Intermediate Switch Nodes should have information about the  
1867 aggregation capability in their switching table and shall not aggregate packets when it is known that next  
1868 level Switch Node does not support this feature.

1869 Devices that implement packet aggregation shall ensure that the size of the MSDU comprising the aggregates  
1870 does not exceed the maximum capacity of the most robust transmission scheme of a PHY burst. The most  
1871 robust transmission scheme refers to the most robust combination of modulation scheme, convolutional  
1872 coding and repetition coding.

#### 1873 **4.3.7.2 Packet aggregation when switching**

1874 Switch Nodes maintain information on the packet aggregation capability of all entries in their switching table,  
1875 i.e. of all switches that are connected to the Subnetwork through them. This capability information is then  
1876 used during traffic switching to/from the connected Switch Nodes.



1877 The packet aggregation capability of a connecting Switch Node is registered at each transit Switch Node at  
1878 the time of its promotion by sniffing relevant information in the PRO\_ACK message.

- 1879 • If the PKT.SID in a PRO\_ACK message is the same as the switching Node, the Node being promoted is  
1880 connected directly to the said Switch Node. The aggregation capability of this new Switch Node is  
1881 registered as the same as indicated in PKT.NAD of the PRO\_ACK packet.
- 1882 • If the PKT.SID in a PRO\_ACK message is different from the SID of the switching Node, it implies that  
1883 the Node being promoted is indirectly connected to this Switch. The aggregation capability for this  
1884 new Switch Node will thus be the same as the aggregation capability registered for its immediate  
1885 Switch, i.e. PKT.SID.

1886 Aggregation while switching packets in uplink direction is performed if the Node performing the Switch  
1887 knows that its uplink path is capable of handling aggregated packets, based on capability information  
1888 exchanged during Registration (REG.CAP\_PA field in REG\_RSP message).

1889 Downlink packets are aggregated by analyzing the following:

- 1890 • If the PKT.SID is the same as the switching Node, then it is the last switching level and the packet will  
1891 arrive at its destination. In this case, the packet may be aggregated if PKT.NAD=0.
- 1892 • If the PKT.SID is different, this is not the last level and another Switch will receive the packet. The  
1893 information of whether or not the packet could be aggregated should be extracted from the switching  
1894 table.

## 1895 **4.3.8 Security**

### 1896 **4.3.8.1 General**

1897 The security functionality provides the MAC layer with confidentiality, authentication, integrity and  
1898 protection against reply attacks through a secure connection method and a key management policy. All  
1899 packets must use the negotiated security profile.

### 1900 **4.3.8.2 Security Profiles**

1901 Several security profiles are provided for managing different security needs, which can arise in different  
1902 network environments. This version of the specification lists three security profiles and leaves scope for  
1903 adding another security profile in future versions.

#### 1904 **4.3.8.2.1 Security Profile 0**

1905 Communications having Security Profile 0 are based on the transmission of MAC SDUs without encryption.  
1906 This profile may be used in application scenarios where either sufficient security is provided by upper  
1907 communication layers or where security is not a major requirement for application use-case.

#### 1908 **4.3.8.2.2 Security Profile 1 and 2**

##### 1909 **4.3.8.2.2.1 General**

1910 Security Profile 1 and 2 are based on several cryptographic primitives, all based upon AES-128, which provides  
1911 secure functionalities for key derivation, key wrapping/unwrapping and authenticated encryption of packets.  
1912 These profiles are specified with the aim of fulfilling all security requirements:"

- 1913 • Confidentiality, authenticity and integrity of packets are guaranteed by the use of an authenticated encryption algorithm.
- 1914
- 1915 • Authentication is guaranteed by the fact that each Node has its own unique key known only by the
- 1916 Node itself and the Base Node.
- 1917 • Replay Attacks are prevented through the use of a message counter of 4 bytes.

#### 1918 **Note:**

1919 The scope of the Security Profile does not address any implementation specific security requirements such  
 1920 as protection against side channel attacks (timing attacks, power attacks, electromagnetic attacks, fault  
 1921 attacks, etc...). The implementer of the security profile needs to assure the cryptographic functionality is  
 1922 adequately protected.

- 1923 • The implementer might consider counter measures depending on the environment PRIME is
- 1924 used. This could include the implementation of an AES algorithm with mitigation for non-invasive
- 1925 attacks (e.g. power analysis or electromagnetic side channel attacks). Additional tamper
- 1926 protection and hardening mechanisms are specified in FIPS 140-3 levels 3 and 4.

#### 1927 **4.3.8.2.2.2 Authenticated Encryption**

1928 The cryptographic algorithms used in this specification are all based on AES, as specified in [16]. The  
 1929 specification describes the algorithm with three possible key sizes. PRIME uses a key length of 128 bit. A key  
 1930 length of 128 bit represents a good level of security for preserving privacy up to 2030 and beyond, as specified  
 1931 in SP800-57 [17], page 66, table 4.

1932 AES is used in CCM mode, as specified in [25]. It is a dual-pass authenticated encryption mode. In the context  
 1933 of this security profile it is used accordingly to the following settings (using the same notations of [25]):

- 1934 •  $n$ : the octet length of the nonce is set to 13. This allows for a maximum message size of 65535
- 1935 bytes.
- 1936 •  $q$ : the octet length of the binary representation of the octet length of the payload is set to 2.
- 1937 •  $t$ : the octet length of the MAC is set to 6. Therefore  $Tlen$ , the MAC bit size, is set to 48.

#### 1938 **4.3.8.2.2.2.1 Key update frequency**

1939 Security profiles set the value of the AES-CCM authentication tag ( $Tlen$ ) to 48 bits. The maximum time limit  
 1940 between two re-keying events for WK and SWK is the value contained in the PIB *MACUpdateKeysTime*. This  
 1941 PIB's maximum allowed value shall be the one defined in Table 15 according to the available number of  
 1942 channels.

1943 **Table 15 - Values of *MACUpdateKeysTime* for different number of channels**

| Available number of channels | Life time in days |
|------------------------------|-------------------|
| 1 x 64Kb/s channel           | 49 days           |
| 2 x 64Kb/s channel           | 24 days           |
| 3 x 64Kb/s channel           | 16 days           |

|                    |         |
|--------------------|---------|
| 4 x 64Kb/s channel | 12 days |
| 5 x 64Kb/s channel | 10 days |
| 6 x 64Kb/s channel | 8 days  |
| 7 x 64Kb/s channel | 7 days  |
| 8 x 64Kb/s channel | 6 days  |

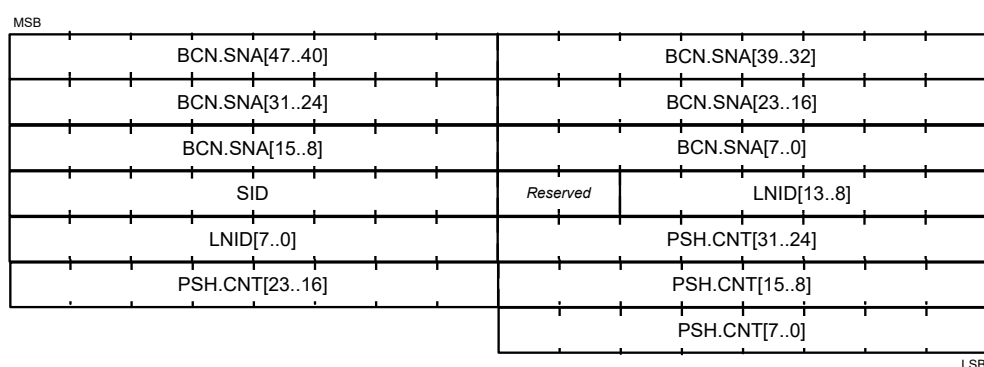
1944

#### 1945 4.3.8.2.2.2 Nonce creation

1946 The nonce is a value used by AES-128-CCM and is required to be unique for each different message that is  
 1947 processed under the same key. In order to maintain this property and to have protections against replay  
 1948 attacks, each Service Node needs to have a 32-bit message counter.

1949 The 13-byte nonce, for each message, is shown in Figure 42 and is composed by the concatenation of the  
 1950 following entities:

- 1951 • 48-bit Subnetwork Address (found in BCN.SNA)
- 1952 • 8-bit SID address, identifying the Switch Node of the Service Node which generated the packet
- 1953 • 2-bit set to 0 for this version of the specification. Reserved for future use.
- 1954 • 14-bit LNID address, identifying the Service Node that generated the packet. The pair SID and  
 1955 LNID should provide a unique address within the subnetwork.
- 1956 • 32-bit Message Counter, number of messages sent by the Service Node which originated the  
 1957 message



1958

1959

Figure 42 – Nonce structure

1960 The nonce SID and LNID entities are derived from the senders SID and LNID as shown in Table 16.

1961

Table 16 –Nonce SID/LNID Derivation

| Packet Type /<br>Affected Keys | Packet Direction | Nonce Entity | Nonce Entity Value |
|--------------------------------|------------------|--------------|--------------------|
|                                | Downlink         | SID          | 0                  |
|                                |                  | LNID         | 0                  |

| Packet Type /<br>Affected Keys                  | Packet Direction            | Nonce Entity | Nonce Entity Value |
|-------------------------------------------------|-----------------------------|--------------|--------------------|
| Unicast<br><i>Key: WK or SWK not re-encoded</i> | <i>Uplink</i>               | SID          | PKT.SID            |
|                                                 |                             | LNID         | PKT.LNID           |
| Switch Re-encoded<br><i>Key: SWK *</i>          | <i>Downlink/<br/>Uplink</i> | SID          | Switch SSID        |
|                                                 |                             | LNID         | 0                  |
| Broadcast/Multicast<br><i>Key: SWK</i>          | <i>Downlink</i>             | SID          | PKT.SID            |
|                                                 |                             | LNID         | 0                  |
|                                                 | <i>Uplink</i>               | SID          | PKT.SID            |
|                                                 |                             | LNID         | PSH.LNID           |

\* Switch re-encoded packets are all ALV messages sent by switches as well as all downlink messages encrypted with SWK. A switch shall know the immediate switch for each SID below it to be able to derive the nonce.

#### 4.3.8.2.2.2.1 Message counter (PSH: CNT)

In order to avoid repetition attacks, in case the message counter of a message is not correctly validated according to this chapter, the message shall be discarded.

In the case of messages protected with WK, each node has a message counter starting from zero, incrementing after each protected message sent. This counter shall be reset after updating to a newer key.

In the case of messages protected with SWK, the counter used for nonce creation shall act following a distributed algorithm. This counter shall also be reset after updating to a newer key.

Upon reception of downlink message, every node shall validate that the counter is bigger than the last downlink message received. If the node is an intermediate Switch Node, once validated, the message shall be re-encoded before switching it down.

Upon reception of uplink message, a Switch Node shall not perform a validation of the CNT except for the cases when the message is processed by them. These are the use cases when it shall validate the message:

- ALV messages
- Data uplink messages switched down by a Direct Switch (does not apply to the intermediate Switch Nodes, only the Direct Switch itself)

A node shall maintain a transmission counter with the next value to be used in CNT field. This value shall be used when the node encodes and re-encodes a message. After using it, the node shall increase its value by 1.

When the node receives a counter bigger than the one stored for transmission, the transmission counter shall be updated with the received counter value + 1. This applies to both downlink and uplink messages. As a guideline, note that this forces any response message to have a counter bigger than the related request message, easing validation requirement and corner cases.

In security profile 1, direct switches are allowed to ignore validation of the first uplink data message from each peer, after that first message, the switch shall perform a validation of all uplink messages. This is to simplify implementation of direct switches on both memory usage and complexity.

1989 Base Node shall validate all the uplink counters, considering all the rules described previously. In order to  
1990 perform this, it shall keep track of individual counters for each node. The algorithm to do so is left up to the  
1991 manufacturer of the Base Node.

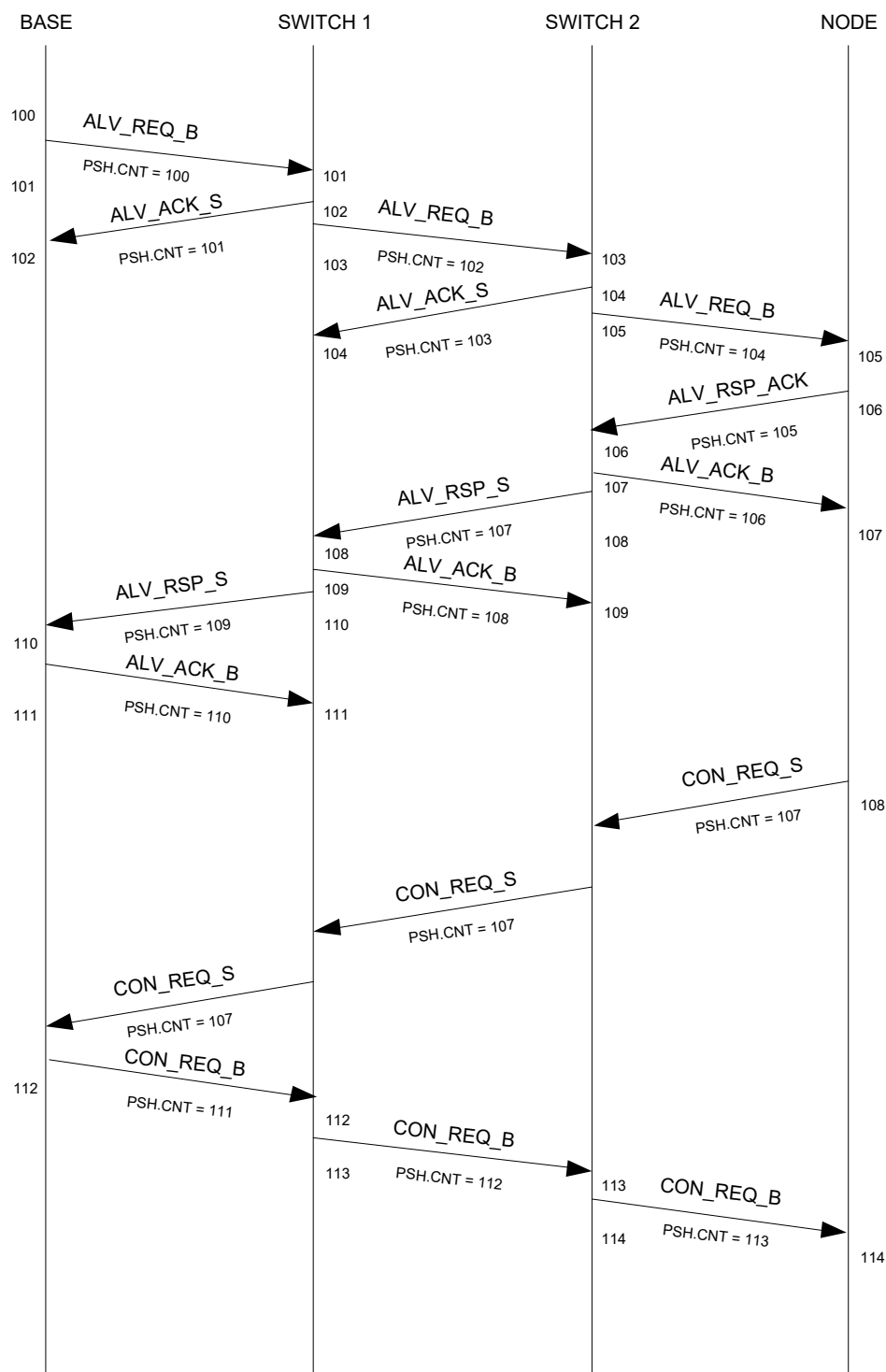


Figure 43 – Counter handling in ALV and request/response messages

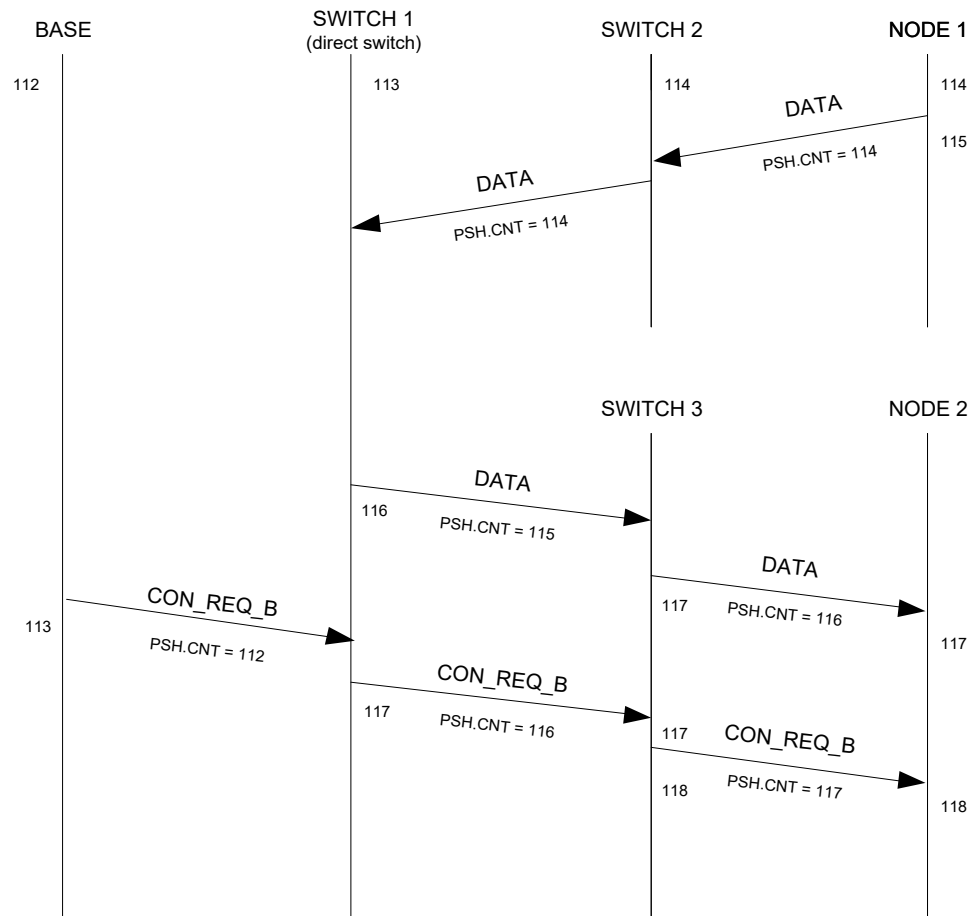


Figure 44 – Counter handling in direct connections and request messages

**4.3.8.2.2.3 Creation of Challenge nonce for REG PDU-s**

Each REG message relies on a 64 bit random number and the Subnetwork Address as a challenge. This challenge shall be used as the nonce for the authentication of the REG\_REQ, REG\_RSP and REG\_REJ messages. The usage sequence is described in detail in 4.6.1.2.

The nonce, for each message, is shown in Figure 45, and is composed by the concatenation of the following entities:

- 40 least significant bits of the Subnetwork Address (found in BCN.SNA)
- For REG\_REQ, Service Node shall generate a 64-bit random number and shall encode that value in the PSH.CNT and REG.CNT fields
- For REG\_RSP and REG\_REJ, REG.CNT shall be the copy of the same field coming from REG\_REQ and PSH.CNT shall be PSH.CNT coming from REG\_REQ incremented by 1, overflowing if necessary

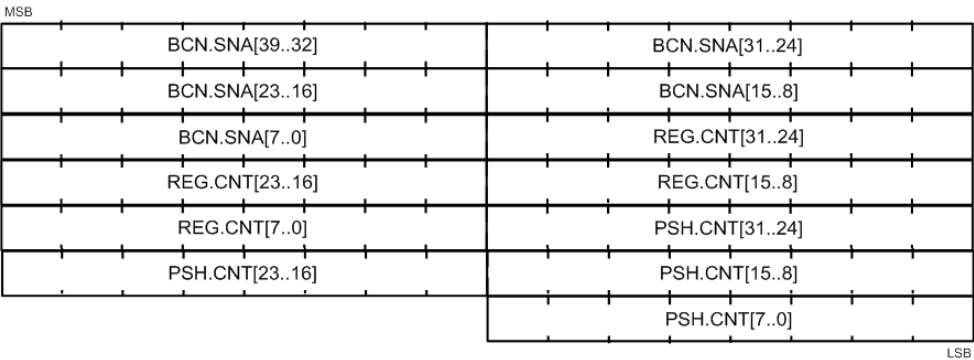


Figure 45 – REG Nonce structure

4.3.8.2.2.3 Key Derivation Algorithm

The method for key derivation is KDF in counter mode as specified in [23] using AES-CMAC [24] with key size of 128 as underlying PRF. This KDF requires 5 values as input:

- $K_i$  which is the master key used to derive the output key  $K_o$
- *Label* which is a string, fixed for the purpose of this security profile at “PRIME\_MAC”
- *Context*, which is a string assuming different values accordingly to the purpose of the output key, which will be described in section 4.3.8.3.2
- $L$  which is the size of the output key, which for the purpose of this security profile is fixed to 128.
- $r$  which is an integer indicating the lengths of the binary representation of the counter and of  $L$ , which is fixed in this security profile to 32

4.3.8.2.2.4 Key Derivation Hierarchy

Figure 46 outlines the Key Derivation hierarchy and the process to derive the Key Wrapping Key (KWK) and the Registration Key (REGK).

The KWK is used to wrap the individual Working Key (WK) and the Subnetwork Working Key (SWK) when sent down from the Base Node to the Terminal Node, while the REGK is used for authentication in the registration process to authenticate both, BN and TN.

The random number generator used for WK and SWK should be compliant with [27].

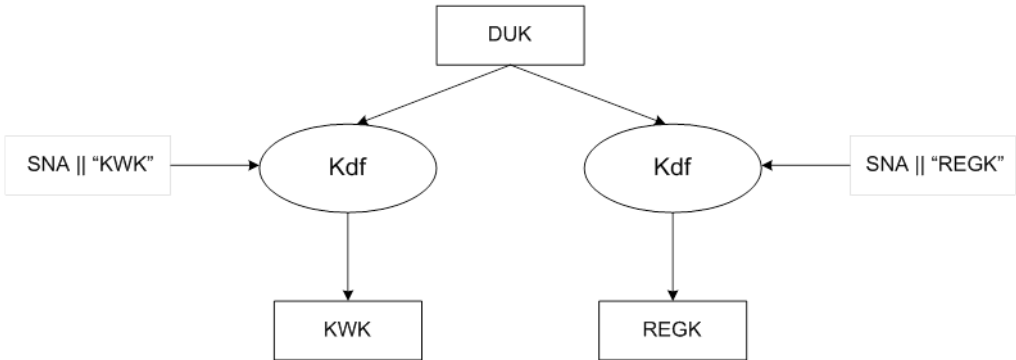


Figure 46 – Key derivation hierarchy

#### 4.3.8.2.2.5 Key Wrapping Algorithm

The method for wrapping and unwrapping keys is referred to AES-128-KW, it is described as KW in [26], and uses AES-128 as underlying cipher. It is used to transmit keys in an encrypted form. In this security profile all keys are of 128 bit, which means that wrapped keys are 192 bits.

#### 4.3.8.2.3 Encryption/Authentication by PDU Types

The following table shows which PDU-s are authenticated (A) and/or encrypted (E) on each of the security profiles. This table shows packet types with their names as presented in section 4.3.8.6. The packet nomination follows the following rules: if the packet is a generic name (e.g. REG), the profile will apply for all the subpacket types not listed in the table (e.g. REG\_ACK).

**Table 17 – Encryption/Authentication by PDU Types**

| PDU Type                                                                                                        | Profile 1 | Profile 2 |
|-----------------------------------------------------------------------------------------------------------------|-----------|-----------|
| REG_REQ, REG_RSP                                                                                                | REGK (A)  | REGK (A)  |
| REG_REJ                                                                                                         | Plain     | REGK (A)  |
| Unicast DATA                                                                                                    | WK (AE)*  | WK (AE)*  |
| SEC                                                                                                             | WK(AE)    | WK(AE)    |
| Multicast DATA, Broadcast DATA, Direct connection DATA                                                          | SWK (AE)* | SWK (AE)* |
| PRO, MUL, CFP, CON, FRA, PCC, ALV, PRO_ACK, MUL_JOIN_B, MUL_LEAVE_S, PRO_DEM_S, PRO_DEM_B, REG_UNR_B, REG_UNR_S | Plain     | SWK (AE)  |
| REG_ACK                                                                                                         | Plain     | WK(A)     |

The rows highlighted with an asterisk (\*) can be optionally sent not encrypted as described in section 4.3.8.6.

#### 4.3.8.3 Negotiation of the Security Profile

##### 4.3.8.3.1 General

All MAC data, including signaling PDUs (all MAC control packets defined in section 4.4.2.6) use the same security profile. This profile is negotiated during the device Registration. In the REG\_REQ message the Terminal indicates a security profile it is able to support in the field REG.SPC. The Base Node may accept this security profile and so accept the Registration, sending back a REG\_RSP with the same REG.SPC value. The Base Node may also accept the Registration, however it sets REG.SPC to 0, 1 or 2 indicating that security profile 0, 1 or 2 is to be used. Alternatively, the Base Node may reject the Registration if the Terminal does not provide an acceptable security profile.

It is recommended that the Terminal first attempts to register using the highest security profile it supports. In case the Base Node replies with a different value for REG.SPC, corresponding to a profile with lower security, the Terminal could refuse the registration by not sending the REG\_ACK. The policy used by the Terminal to refuse a registration with a lower than expected security profile is out of the scope of this specification.



---

#### 2054 4.3.8.3.2 Key Types and Key Hierarchy

2055 The key hierarchy of Security Profile 1 and 2 is based on three assumptions:

- 2056 1. There is a 128 bit unique key on each service node called Device Unique Key (DUK). How this key is  
2057 generated, provided to service nodes, is out of the scope of this specification. The DUK is managed  
2058 by macSecDUK (refer to section 6.2.3.5.1.)  
2059 2. The Base Node must have knowledge of a Service Node's DUK by only knowing its EUI-48.  
2060 3. As specified by [REF TO NIST SP800-57Part1] "In general, a single key should be used for only one  
2061 purpose".

2062 The keys and their respective usage are:

2063 **Device Unique Key (DUK):** DUK is used only for key derivation purposes, using the KDF described in section  
2064 4.3.8.2.2.3. It has the requirement to be unique for each device. It is used to generate KWK and REGK.

2065 **Key Wrapping Key (KWK):** This key is derived from DUK using the concatenation of the Subnetwork Address  
2066 (SNA) and the string "KWK" as *Context*. It is used to unwrap the keys received from the Base Node.

2067 **REG Key (REGK):** This key is derived from DUK using the concatenation of the Subnetwork Address (SNA) and  
2068 the string "REGK" as *Context*. It is used to protect, through AES-128-CCM, some of the REG control messages,  
2069 specifically it is used for: REG\_REQ, REG\_RSP, REG\_REJ only when REG.R=0. The reason is that there hasn't  
2070 been any communication with the Base Node yet, so no other shared keys have been established.

2071 **Working Key (WK):** This key is used to encrypt all the unicast data that is transmitted from the Base Node to  
2072 a Service Node and vice versa. Each registered Service Node would have a unique WK that is known only to  
2073 the Base Node and itself. The WK is randomly generated by the Base Node, wrapped through AES-128-KW  
2074 and transmitted by the Base Node in REG\_RSP and SEC messages.

2075 **Subnetwork Working Key (SWK):** The SWK is shared by the entire Subnetwork. The SWK is randomly  
2076 generated by the Base Node, wrapped through AES-128-KW and transmitted by the Base Node in REG\_RSP  
2077 and SEC messages.

2078 The WK and the SWK have a limited validity time related to the random sequence generation period. The  
2079 random sequence is regenerated and distributed by the Base Node at least every *MACUpdateKeysTime*  
2080 seconds through the SEC control packet. If a device does not receive a new SEC message within  
2081 *MACUpdateKeysTime* it shall move back from its present functional state to a *Disconnected* functional state.

2082 The key hierarchy has been designed to ensure security of the required MAC keys, to follow NIST  
2083 specifications and to be as simple as possible.

#### 2084 4.3.8.4 Key Distribution and Management

2085 The Security Profile for data traffic is negotiated when a device is registered. The REG control packet contains  
2086 specific fields to indicate the Security Profile for respective devices. All connections to/from the device would  
2087 be required to follow the Security Profile negotiated at the time of Registration. There cannot be a difference  
2088 in Security Profile across multiple connections involving the same device. The only exception to this would  
2089 be the Base Node.

2090 All keys are never transmitted in non-encrypted form over the physical channel. The SEC unicast messages  
2091 transmitted by the Base Node at regular intervals contain random keys for both unicast and non-unicast  
2092 traffic.

2093 When a device initially registers on a Subnetwork, the REG response from the Base Node contains the  
2094 wrapped SWK and WK. If the SN cannot unwrap the keys successfully, it shall discard the REG response and  
2095 continue trying to register.

2096 The process of updating WK is as follows:

- 2097 • The SN shall start using the new WK immediately for transmission.
- 2098 • The BN shall use the old WK for transmission until receiving the SEC response.
- 2099 • The SN and BN shall be able to receive messages encrypted with both new and old WK until SEC  
2100 exchange completes or Control packet retransmission (see 4.4.2.6.2) completes.
- 2101 • If the SN cannot unwrap the new WK successfully, it shall indicate that it could not update WK and  
2102 shall continue using the old WK. In such case, the BN shall retransmit the SEC message.

2103 Upon reception of a new SWK that is successfully unwrapped, the node shall maintain the old SWK and use  
2104 it to decrypt and encrypt the appropriate messages until:

- 2105 1. a message is received encrypted with the new SWK.
- 2106 2. the expiration of a 3-hour timer.

2107 The timer provides a method for ensuring the old SWK will not be used indefinitely.

2108 If the SN cannot unwrap the new SWK successfully, it shall indicate that it could not update SWK and shall  
2109 continue using the old SWK. In such case, the BN shall retransmit the SEC message.

2110 It is recommended that a Base Node register a Terminal Node with the old SWK and immediately perform a  
2111 SEC procedure, with the registering Terminal Node, if the Terminal Node registers during an in progress SWK  
2112 SEC procedure.

2113 The Base Node shall maintain the old SWK for duration not to exceed 3-hours from the beginning of the SWK  
2114 SEC procedure. This limit constrains the SWK SEC procedure duration to 3-hours. The Base Node can stop  
2115 accepting the old SWK from any node before that duration, e.g. if it is certain that a Service Node has received  
2116 a packet with the new SWK.

#### 2117 **4.3.8.5 Encryption and Authentication**

##### 2118 **4.3.8.5.1 Security Profile 0**

2119 Not Applicable.

##### 2120 **4.3.8.5.2 Security Profile 1 and 2**

2121 Security Profiles 1 and 2 make use of AES-CCM for packet protection but there are three different cases,  
2122 accordingly to section 4.3.8.2.3:

- 2123 • Plain: in the case the packet is not processed by AES-CCM and there is no Tag

Authentication Only: in this case the packet header where the PKT.RM is masked with 0, PSH, TREF (if exists) where the values are masked with 0, ARQ (if exists) and the payload should be processed by AES-CCM as associated data. This situation is depicted in Figure 47.

- Authentication and Encryption: in this case the packet header where the PKT.RM is masked with 0, PSH, TREF (if exists) where the values are masked with 0 and ARQ (if exists) should be processed as associated data, thus only being authenticated, while the payload should be processed as payload, thus authenticated and encrypted. This situation is depicted in Figure 48.

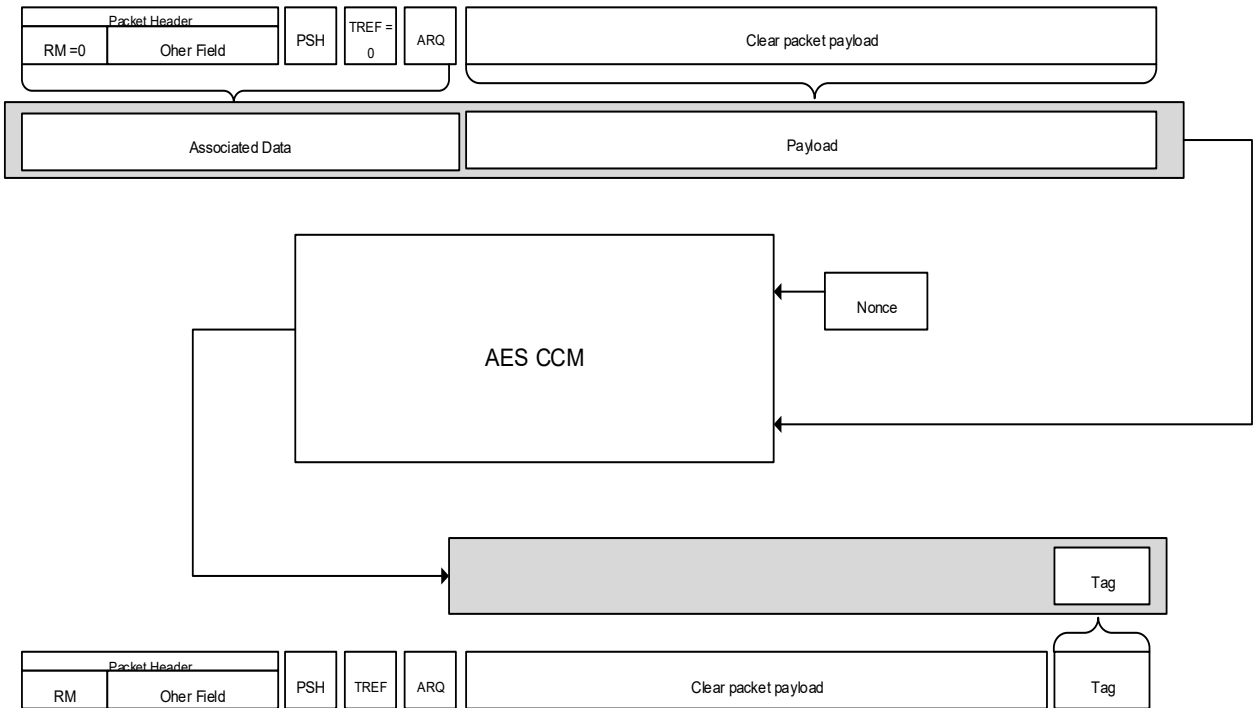


Figure 47 – Security profile 1 and 2 encryption algorithm (authentication only)

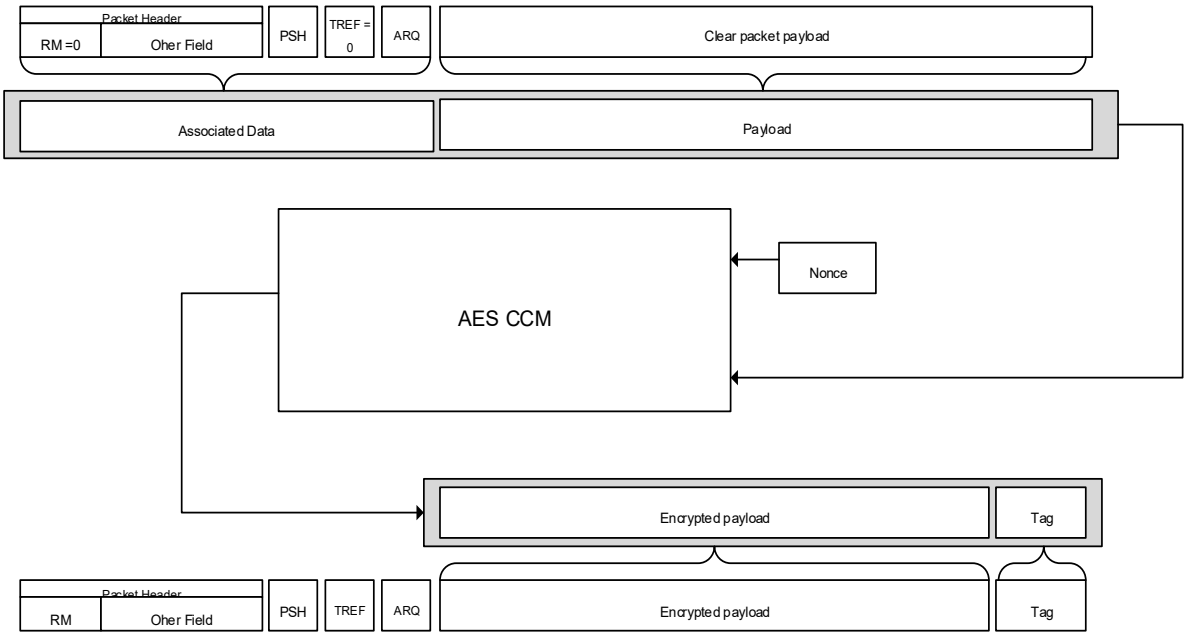


Figure 48 – Security profile 1 and 2 encryption algorithm (authentication and encryption)

#### 2135 **4.3.8.6 Unicast and Multicast connection security negotiation**

2136 There are some use cases in which it is not desirable to authenticate and encrypt data packets. One use case  
2137 of this is the firmware upgrade for images that are already signed and potentially encrypted.

2138 For such cases both the unicast and multicast connection establishment procedures have a feature that  
2139 allows a negotiation for enabling or disabling the security on DATA packets. This procedure only applies to  
2140 profiles 1 and 2, because profile 0 does not allow encryption.

2141 Each side shall indicate in the CON\_REQ and MUL\_JOIN packets if the DATA packets for that connection shall  
2142 be authenticated and encrypted or not.

- 2143 • In case both sides indicate that the DATA packets shall not be securitized, then the packets shall be  
2144 sent without being authenticated or encrypted.
- 2145 • If at least one of the sides indicates that the packets shall be securitized, then the packets shall be  
2146 authenticated and encrypted.

2147 Regarding broadcast connections, no negotiation takes place. Broadcast connections shall always  
2148 be authenticated and encrypted.

#### 2149 **4.3.8.7 Unicast and Multicast connection security negotiation**

2150 After initial negotiation, any multicast connection sending plain data can be upgraded to be  
2151 authenticated and encrypted at any point. The usual use case is that a new node has joined the  
2152 multicast group negotiating the connection to be authenticated and encrypted.

2153 The upgrade mechanism is to start sending multicast data authenticated and encrypted, without  
2154 any additional control message. Any node receiving multicast data that is authenticated and  
2155 encrypted shall assume the connection has been upgraded to be authenticated and encrypted  
2156 from the reception of that data.

2157 There is no procedure for a multicast connection to be downgraded to send plain data.

### 2158 **4.4 MAC PDU format**

#### 2159 **4.4.1 General**

2160 There are different types of MAC PDUs for different purposes.

2161 Note that reserved bits must always be set to 0.

2162 To allow for PDU compatibility with previous specification versions, when it is necessary to include an  
2163 additional field in an existing PDU, several options are possible:

- 2164 1) If the new field fits in the reserved bits available in the PDU and the null value of the new field does  
2165 not interfere with the expected behaviour of the reserved bits, it can be included using them.
- 2166 2) If the new field requires more bits than the reserved bits available in the PDU, a reserved bit shall be  
2167 used to indicate the presence of the new field. The new field may be added using other reserved bits

– as long as the null value does not interfere with the expected behaviour of the reserved bits – or, if that is not possible, it will be added at the end of the PDU.

- 3) If some fields are not used in the PDU, even if they are not reserved, a reserved bit may change their meaning.

## 4.4.2 Generic MAC PDU

### 4.4.2.1 General

Most Subnetwork traffic comprises Generic MAC PDUs (GPDU). GPDUs are used for all data traffic and most control traffic. All MAC control packets are transmitted as GPDUs.

GPDU composition is shown in Figure 49. It is composed of a Generic MAC Header followed by one or more MAC packets and 32 bit CRC appended at the end.



Figure 49 – Generic MAC PDU format

### 4.4.2.2 Generic MAC Header

The Generic MAC Header format is represented in Figure 46 and Table 18. The size of the Generic MAC Header is 3 bytes. Table 18 enumerates each field of a Generic MAC Header.

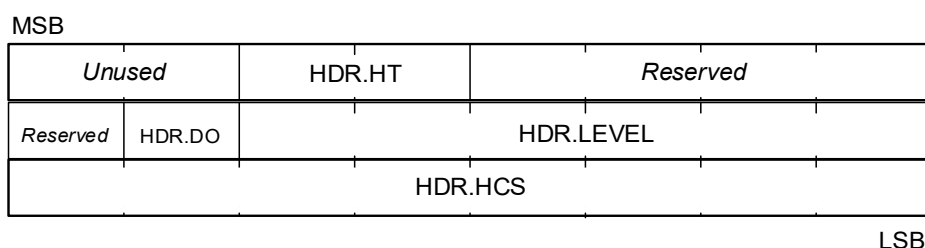


Figure 50 - Generic MAC header

Table 18 - Generic MAC header fields

| Name            | Length | Description                                                                                                                                         |
|-----------------|--------|-----------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>Unused</i>   | 2 bits | Unused bits those are always 0; included for alignment with MAC_H field in PPDU header (Section 3.3.2.3).                                           |
| HDR.HT          | 2 bits | Header Type.<br>HDR.HT = 0 for GPDU                                                                                                                 |
| <i>Reserved</i> | 5 bits | Always 0 for this version of the specification. Reserved for future use.                                                                            |
| HDR.DO          | 1 bit  | Downlink/Uplink. <ul style="list-style-type: none"> <li>HDR.DO=1 if the MAC PDU is Downlink.</li> <li>HDR.DO=0 if the MAC PDU is uplink.</li> </ul> |

| Name      | Length | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |
|-----------|--------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| HDR.LEVEL | 6 bits | Level of the PDU in switching hierarchy.<br>The packets between the level 0 and the Base Node are of HDR.LEVEL=0. The packets between levels k and k-1 are of HDR.LEVEL=k. <ul style="list-style-type: none"><li>• If HDR.DO=0, HDR.LEVEL represents the level of the transmitter of this packet.</li><li>• If HDR.DO=1, HDR.LEVEL represents the level of the receiver of this packet.</li></ul>                                                                                                                                                                                                                                                                                                                                                                                                 |
| HDR.HCS   | 8 bits | Header Check Sequence.<br>A field for detecting errors in the header and checking that this MAC PDU is from this Subnetwork. The transmitter shall calculate the CRC of the SNA concatenated with the first 2 bytes of the header and insert the result into the HDR.HCS field (the last byte of the header). The concatenation is performed such that the SNA is the first input to the divisor. The CRC shall be calculated as the remainder of the division (Modulo 2) of the polynomial $M(x) \cdot x^8$ by the generator polynomial $g(x)=x^8+x^2+x+1$ . $M(x)$ is the input polynomial, which is formed by the bit sequence of the concatenation of the SNA and the header excluding the HDR.HCS field, and the msb of the bit sequence is the coefficient of the highest order of $M(x)$ . |

4.4.2.3 Packet structure

A packet is comprised of a Packet Header and Packet Payload. Figure 51 shows the structure.

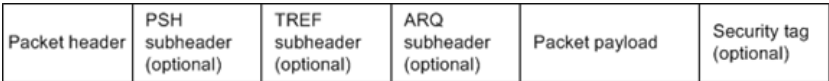


Figure 51 - Packet structure

Packet header is 7 bytes in length and its composition is shown in Figure 52. Table 19 enumerates the description of each field.

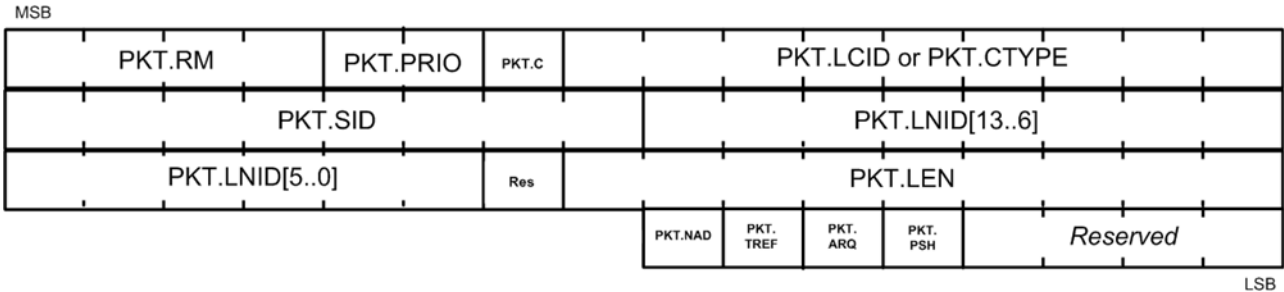


Figure 52 – Packet Header

To simplify, the text contains references to the PKT.NID fields as the composition of the PKT.SID and PKT.LNID. The field PKT.CID is also described as the composition of the PKT.NID and the PKT.LCID. The composition of these fields is described in Figure 53.

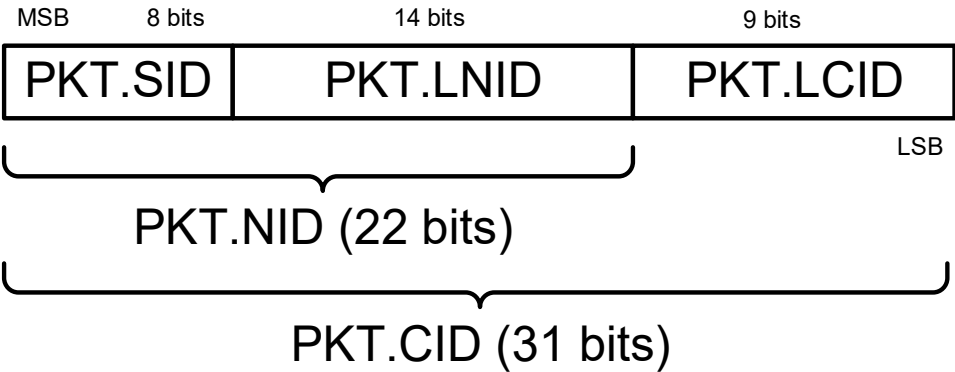


Figure 53 - PKT.CID structure

Table 19 – Packet header fields

| Name                    | Length | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                       |
|-------------------------|--------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| PKT.RM                  | 4 bits | Weakest modulation this node can decode from the receiving peer.<br>If the PCH is PLC <ul style="list-style-type: none"><li>0 – DBPSK</li><li>1 – DQPSK</li><li>2 – D8PSK</li><li>3 – Not used</li><li>4 – DBPSK + Convolutional Code</li><li>5 – DQPSK + Convolutional Code</li><li>6 – D8PSK + Convolutional Code</li><li>7-11 – Not used</li><li>12 – Robust DBPSK</li><li>13 – Robust DQPSK</li><li>14 – Not used</li><li>15 – Outdated information</li></ul> |
| PKT.PRIO                | 2 bits | Indicates packet priority between 0 and 3.                                                                                                                                                                                                                                                                                                                                                                                                                        |
| PKT.C                   | 1 bits | Control <ul style="list-style-type: none"><li>If PKT.C=0 it is a data packet.</li><li>If PKT.C=1 it is a control packet.</li></ul>                                                                                                                                                                                                                                                                                                                                |
| PKT.LCID /<br>PKT.CTYPE | 9 bits | Local Connection Identifier or Control Type <ul style="list-style-type: none"><li>If PKT.C=0, PKT.LCID represents the Local Connection Identifier of data packet.</li><li>If PKT.C=1, PKT.CTYPE represents the type of the control packet.</li></ul>                                                                                                                                                                                                              |

| Name     | Length  | Description                                                                                                                                                                                                                                            |
|----------|---------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| PKT.SID  | 8 bits  | Switch identifier <ul style="list-style-type: none"> <li>If HDR.DO=0, PKT.SID represents the SID of the packet source.</li> <li>If HDR.DO=1, PKT.SID represents the SID of the packet destination.</li> </ul>                                          |
| PKT.LNID | 14 bits | Local Node identifier. <ul style="list-style-type: none"> <li>If HDR.DO=0, PKT.LNID represents the LNID of the packet source</li> <li>If HDR.DO=1, PKT.LNID represents the LNID of the packet destination.</li> </ul>                                  |
| Reserved | 1bit    | Always 0 for this version of the specification. Reserved for future use.                                                                                                                                                                               |
| PKT.LEN  | 9 bits  | Length of the packet excluding the packet header and the authentication tag (if present). It is the sum of the lengths of the payload and the subheaders (if any).                                                                                     |
| PKT.NAD  | 1 bit   | No Aggregation at Destination <ul style="list-style-type: none"> <li>If PKT.NAD=0 the packet may be aggregated with other packets at destination.</li> <li>If PKT.NAD=1 the packet may not be aggregated with other packets at destination.</li> </ul> |
| PKT.TREF | 1 bit   | TREF subheader presence: <ul style="list-style-type: none"> <li>If PKT.TREF=0 the packet doesn't include a TREF subheader.</li> <li>If PKT.TREF=1 the packet includes a TREF subheader.</li> </ul>                                                     |
| PKT.ARQ  | 1 bit   | ARQ subheader presence: <ul style="list-style-type: none"> <li>If PKT.ARQ=0 the packet doesn't include an ARQ subheader.</li> <li>If PKT.ARQ=1 the packet includes an ARQ subheader.</li> </ul>                                                        |
| PKT.PSH  | 1 bit   | Packet security subheader presence: <ul style="list-style-type: none"> <li>If PKT.PSH=0 the packet doesn't include a security subheader.</li> <li>If PKT.PSH=1 the packet includes a security subheader.</li> </ul>                                    |
| Reserved | 4 bits  | Always 0 for this version of the specification. Reserved for future use.                                                                                                                                                                               |

2206

2207 The "ARQ subheader", "TREF subheader" and "security subheader" are optional. Their presence depends on  
2208 the PKT.ARQ, PKT.TREF and PKT.PSH flags. The description of the ARQ subheader will be done in the section  
2209 4.7.3.2 and the description of the TREF subheader will be done in section 4.8. MAC Control packets shall not  
2210 include a TREF or ARQ subheader.

#### 2211 4.4.2.4 CRC

2212 The CRC is the last field of the GPDU. It is 32 bits long. It is used to detect transmission errors. The CRC shall  
2213 cover the concatenation of the SNA with the GPDU except for the CRC field itself. The concatenation is  
2214 performed with SNA (48-bit Subnetwork Address) first such that the input to the CRC divisor starts with the



most significant bit of the SNA, after shifting the 6 byte SNA the GPDU follows starting with the first bit of the GPDU. An example can be seen in Annex A.

The input polynomial  $M(x)$  is formed as a polynomial whose coefficients are bits of the data being checked (the first bit to check is the highest order coefficient and the last bit to check is the coefficient of order zero). The Generator polynomial for the CRC is  $G(x)=x^{32}+x^{26}+x^{23}+x^{22}+x^{16}+x^{12}+x^{11}+x^{10}+x^8+x^7+x^5+x^4+x^2+x+1$ . The remainder  $R(x)$  is calculated as the remainder from the division of  $M(x) \cdot x^{32}$  by  $G(x)$ . The coefficients of the remainder shall then be the resulting CRC.

#### 4.4.2.5 Security header

For the security profiles 1 and 2, the security subheader contains the needed information to authenticate and/or encrypt the packet.

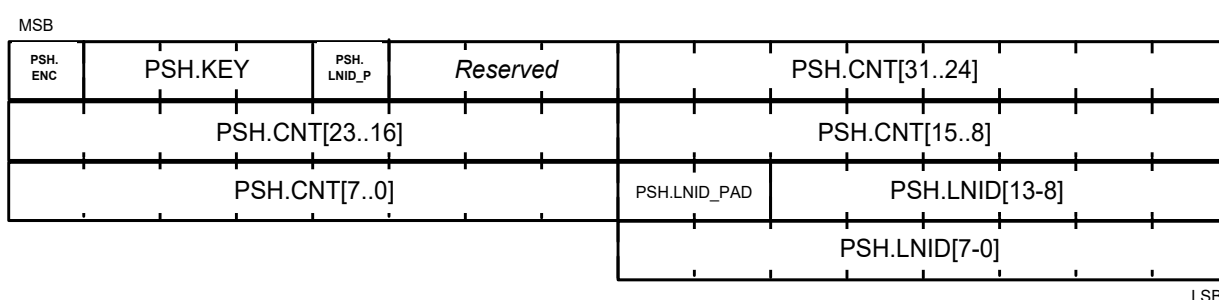


Figure 54 – Security subheader

The description of the fields is described in the following table.

Table 20 – Security subheader fields

| Name       | Length | Description                                                                                                                                           |
|------------|--------|-------------------------------------------------------------------------------------------------------------------------------------------------------|
| PSH.ENC    | 1 bit  | Flag to determine if the packet is encrypted:<br>0 – The packet is Authenticated<br>1 – The packet is Authenticated and Encrypted                     |
| PSH.KEY    | 3 bits | Key used for the encoding of this packet:<br>0 – WK<br>1 – SWK<br>2 – REG<br>3-8 – Reserved for future used.                                          |
| PSH.LNID_P | 1 bit  | Flag to determine if the PSH.LNID is present (counting the reserved bits leading it).<br>0 – If PSH.LNID is not present<br>1 – If PSH.LNID is present |
| Reserved   | 3 bits | Always 0 for this version of the specification. Reserved for future use.                                                                              |

| Name                 | Length  | Description                                                                                                                                                                |
|----------------------|---------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| PSH.CNT              | 32 bits | Counter to be used in the nonce composition.<br><br><b>* For replay protection, receiving node needs to discard packets that do not follow the rules described in 0.</b>   |
| PSH.LNID_P<br>A<br>D | 2 bits  | Always 0 for this version of the specification. Reserved for future use. Only present if PSH.LNID_P field set to one.                                                      |
| PSH.LNID             | 14 bits | Transmitter LNID field to create the nonce when it cannot be derived from the packet. The exception being REG packets.<br><br>Only present if PSH.LNID_P field set to one. |

2229 When the security header is present, a 48-bit authentication tag is appended to the packet. The  
2230 authentication tag is the output of the AES-CCM operation (see Figure 48).

#### 2231 4.4.2.6 MAC control packets

##### 2232 4.4.2.6.1 General

2233 MAC control packets enable a Service Node to communicate control information with their Switch Node,  
2234 Base Node and vice versa. A control packet is transmitted as a GPDU and is identified with PKT.C bit set to 1  
2235 (See section 4.4.2 for more information about the fields of the packets).

2236 There are several types of control messages. Each control message type is identified by the field PKT.CTYPE.  
2237 Table 21 lists the types of control messages. The packet payload (see section 4.4.2.3) shall contain the  
2238 information carried by the control packets. This information differs depending on the packet type.

2239 Table 21 - MAC control packet types

| Type (PKT.CTYPE) | Packet name | Packet description              |
|------------------|-------------|---------------------------------|
| 1                | REG         | Registration management         |
| 2                | CON         | Connection management           |
| 3                | PRO         | Promotion management            |
| 5                | FRA         | Frame structure change          |
| 6                | CFP         | Contention-Free Period request  |
| 7                | ALV         | Keep-Alive                      |
| 8                | MUL         | Multicast Management            |
| 10               | SEC         | Security information            |
| 11               | PCC         | Programmed configuration change |

2240

#### 2241 4.4.2.6.2 Control packet retransmission

2242 For recovery from lost control messages, a retransmit scheme is defined. MAC control transactions  
2243 comprising of exchange of more than one control packet may follow the retransmission mechanism  
2244 described in this section.

2245 The retransmission scheme shall be applied to the following packets when they require a response:

- 2246 • CON\_REQ\_S, CON\_REQ\_B;
- 2247 • CON\_CLS\_S, CON\_CLS\_B;
- 2248 • REG\_RSP;
- 2249 • PRO\_REQ\_B;
- 2250 • MUL\_JOIN\_S, MUL\_JOIN\_B;
- 2251 • MUL\_LEAVE\_S, MUL\_LEAVE\_B;
- 2252 • MUL\_SW\_LEAVE\_B
- 2253 • SEC\_REQ

2254 Devices involved in a MAC control transaction using retransmission mechanism shall maintain a retransmit  
2255 timer and a message fail timer.

2256 At the requester of a control message transaction:

- 2257 • When the one of the above messages in a transaction is transmitted, the retransmit timer is started  
2258 with value greater or equal to macMinCtlReTxTimer and the control message fail timer is started with  
2259 value macCtrlMsgFailTime.

2260 If a response message is received the retransmit timer and control message fail timer are stopped and the  
2261 transaction is considered complete. Note that it is possible to receive further response messages. These  
2262 would be messages that encountered network delays.

- 2263 • If the retransmit timer expires the control message is retransmitted and the retransmit timer is re-  
2264 started with value greater or equal to macMinCtlReTxTimer (value can be different from the previous  
2265 one).
- 2266 • If the control message fail timer expires, failure result corresponding to respective MAC-SAP should  
2267 be returned to the calling entity. Implementations may also choose to inform their local management  
2268 entity of such failure. If the retransmission is done by the Service Node, the device shall return to the  
2269 *Disconnected* functional state

2270 At the responder of a control message transaction:

- 2271 • The receiver of a message must determine itself if this message is a retransmit. If so, no local action  
2272 is needed other than sending a reply to the response.

2273 If the received message is not a retransmit, the message shall be processed and a response returned to the  
2274 sender.

- 2275 • For transactions which use three messages in the transaction, e.g. promotion as shown in 4.6.3, the  
2276 responder shall perform retransmits in exactly the same way as the requester. This ensures that if the  
2277 third message in the transaction is lost, the message shall be retried and the transaction completed.

2278 The following message sequence charts show some examples of retransmission. Figure 55 shows two  
2279 successful transactions without requiring retransmits.

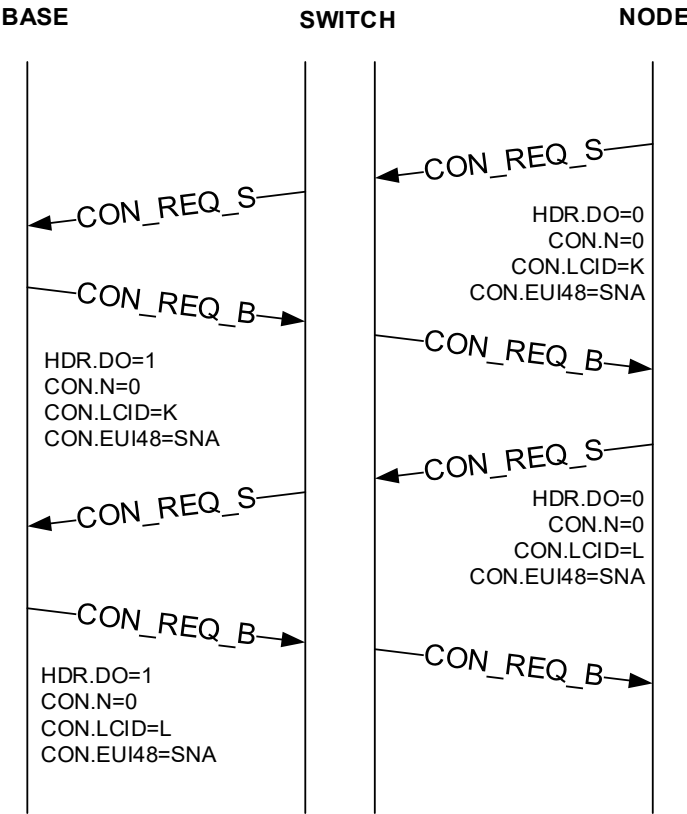


Figure 55 – Two transactions without requiring retransmits

Figure 56 shows a more complex example, where messages are lost in both directions causing multiple retransmits before the transaction completes.

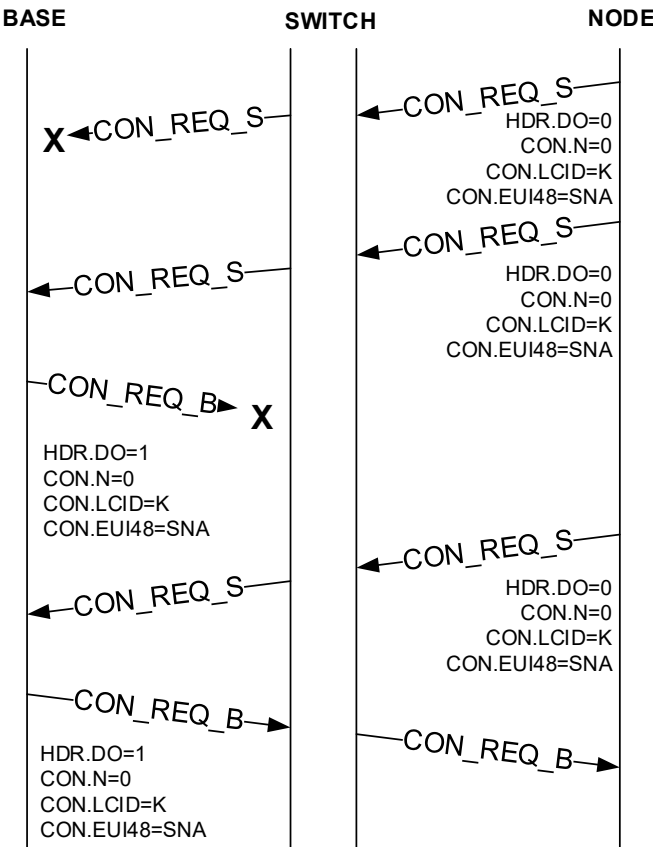


Figure 56 - Transaction with packet loss requiring retransmits

Figure 57 shows the case of a delayed response causing duplication at the initiator of the control transaction.

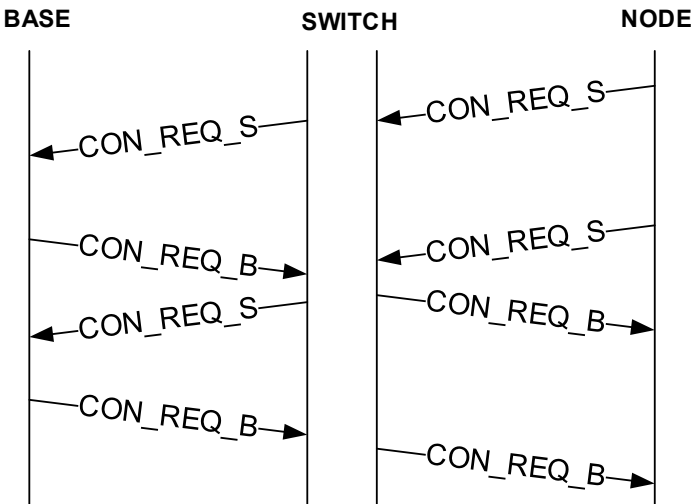


Figure 57 – Duplicate packet detection and elimination

4.4.2.6.3 REG control packet (PKT.CTYPE=1)

This control packet is used to negotiate the Registration process. The description of data fields of this control packet is described in Table 22 and Figure 58. The meaning of the packets differs depending on the direction of the packet. This packet interpretation is explained in Table 22. These packets are used during the registration and unregistration processes, as explained in 4.6.1 and 4.6.2.

Table 22 - REG control packet fields

| Name       | Length | Description                                                                                                                                                                                                                                                                  |
|------------|--------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| REG.N      | 1 bit  | <p>Negative</p> <p><b>REG.N=1 for the negative register;</b></p> <p><b>REG.N=0 for the positive register.</b></p> <p>(see Table 21)</p>                                                                                                                                      |
| REG.R      | 1 bit  | <p>Roaming</p> <p>REG.R=1 if Node already registered and wants to perform roaming to another Switch;</p> <p>REG.R=0 if Node not yet registered and wants to perform a clear registration process.</p> <p>It shall be set by SN in REG_REQ.</p>                               |
| REG.SPC    | 2 bits | <p>Security Profile Capability for Data PDUs:</p> <p>REG.SPC=0 No encryption capability;</p> <p>REG.SPC=1 Security profile 1 capable device;</p> <p>REG.SPC=2 Security profile 2 capable device;</p> <p>REG.SPC=3 Security profile 3 capable device (not yet specified).</p> |
| REG.CAP_R  | 1 bit  | <p>Robust mode Capable</p> <p>1 if the device is able to transmit/receive robust mode frames</p> <p>0 if the device is not</p> <p>It shall be set by SN in REG_REQ</p>                                                                                                       |
| REG.CAP_BC | 1 bit  | <p>Backwards Compatible with 1.3.6</p> <p>1 if the device can operate in backwards compatible mode with 1.3.6 PRIME</p> <p>0 if the device is not</p> <p>It shall be set by SN in REG_REQ and by BN in REG_RSP.</p>                                                          |

| Name        | Length | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |
|-------------|--------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| REG.CAP_SW  | 1 bit  | <p>Switch Capable</p> <p>1 if the device is able to behave as a Switch Node;</p> <p>0 if the device is not.</p> <p>It shall be set by SN in REG_REQ.</p>                                                                                                                                                                                                                                                                                                                                                                                 |
| REG.CAP_PA  | 1 bit  | <p>Packet Aggregation Capability</p> <p>1 if the SN device has packet aggregation capability; if the BN device has packet aggregation capability together with all the switches along the downlink path to the registration requesting SN and the requesting SN itself;</p> <p>0 otherwise.</p> <p>It shall be set by SN in REG_REQ and by BN in REG_RSP.</p>                                                                                                                                                                            |
| REG.CAP_CFP | 1 bit  | <p>Contention Free Period Capability</p> <p>1 if the device is able to perform the negotiation of the CFP;</p> <p>0 if the device cannot use the Contention Free Period in a negotiated way.</p> <p>It shall be set by SN in REG_REQ and by BN in REG_RSP.</p>                                                                                                                                                                                                                                                                           |
| REG.CAP_DC  | 1 bit  | <p>Direct Connection Capability</p> <p>1 if the device is able to perform direct connections;</p> <p>0 if the device is not able to perform direct connections.</p> <p>It shall be set by SN in REG_REQ.</p>                                                                                                                                                                                                                                                                                                                             |
| REG.ALV_F   | 1 bit  | <p>Bit to indicate which ALV mechanism is required to be used by the new Service Node while it is part of this Subnetwork.</p> <p>1 ALV procedure of v1.4 shall be used</p> <p>0 ALV procedure of v1.3.6 (section K.2.5) shall be used.</p> <p>It shall be set by BN in REG_RSP.</p> <p><b>Note:</b> Base Node shall not selectively use different values of this bit between different Service Nodes in its Subnetwork. In case ALV procedure of v1.3.6 is used, all Service Nodes shall be instructed with REG.ALV_F bit set to 0.</p> |

| Name        | Length | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |
|-------------|--------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| REG.CAP_MP  | 1 bit  | <p>Multi-PHY Capability</p> <p>REG_REQ</p> <p>1 if the device is able to generate Multi-PHY promotion message</p> <p>0 if not</p> <p>REG_RSP</p> <p>1 if the base node is able to process Multi-PHY promotion message</p> <p>0 if not</p>                                                                                                                                                                                                                                                                                                                                                                         |
| REG.CAP_ARQ | 1 bit  | <p>ARQ Capable</p> <p>1 if the device is able to establish ARQ connections;</p> <p>0 if the device is not able to establish ARQ connections.</p> <p>It shall be set by SN in REG_REQ and by BN in REG_RSP.</p>                                                                                                                                                                                                                                                                                                                                                                                                    |
| REG.TIME    | 3 bits | <p>Time to wait for an ALV procedure before assuming the Service Node has been unregistered by the Base Node.</p> <p>ALV.TIME = 0 =&gt; 128 seconds ~ 2.1 minutes;</p> <p>ALV.TIME = 1 =&gt; 256 seconds ~ 4.2 minutes;</p> <p>ALV.TIME = 2 =&gt; 512 seconds ~ 8.5 minutes;</p> <p>ALV.TIME = 3 =&gt; 2048 seconds ~ 34.1 minutes;</p> <p>ALV.TIME = 4 =&gt; 4096 seconds ~ 68.3 minutes;</p> <p>ALV.TIME = 5 =&gt; 8192 seconds ~ 136.5 minutes;</p> <p>ALV.TIME = 6 =&gt; 16384 seconds ~ 273.1 minutes;</p> <p>ALV.TIME = 7 =&gt; 32768 seconds ~ 546.1 minutes;</p> <p>It shall be set by BN in REG_RSP.</p> |
| REG.EUI-48  | 48 bit | <p>EUI-48 of the Node</p> <p>EUI-48 of the Node requesting the Registration.</p>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |



| Name         | Length   | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                        |
|--------------|----------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| REG.RM_F     | 2 bits   | <p>Forces an encoding for the given node disabling robustness-management for its transmission, it can be disabled.</p> <p>Only used in REG_RSP. In all other message variants, this shall be 0.</p> <p>0 - Disable, automatic robustness-management by the service nodes</p> <p>1 - DBPSK_CC, device shall transmit always in DBPSK_CC</p> <p>2 - DQPSK_R, device shall transmit always in DQPSK_R</p> <p>3 - DBPSK_R, device shall transmit always in DBPSK_R</p> |
| REG.SAR_SIZE | 3 bits   | <p>Maximum SAR segment size the service node shall use.</p> <p>Only used in REG_RSP. In all other message variants, this shall be 0.0: Not mandated by BN (SAR operates normally)</p> <p>1: SAR = 16 bytes</p> <p>2: SAR =32 bytes</p> <p>3: SAR = 48 bytes</p> <p>4: SAR =64 bytes</p> <p>5: SAR =128 bytes</p> <p>6: SAR =192 bytes</p> <p>7: SAR =255 bytes</p>                                                                                                 |
| REG.PLC_ATT  | 3 bits   | <p>PLC network attenuation field to be used in conjunction with the BCN.PLC_ATT field to obtain a 5-bit attenuation in dB (0-31). This field is only set in the REG_RSP message.</p>                                                                                                                                                                                                                                                                               |
| REG.CNT      | 32 bits  | <p>A counter to be used as the nonce for the registration PDU-s authentication/encryption.</p>                                                                                                                                                                                                                                                                                                                                                                     |
| REG.SWK      | 192 bits | <p>Subnetwork key wrapped with KWK that shall be used to derive the Subnetwork working key</p>                                                                                                                                                                                                                                                                                                                                                                     |
| REG.WK       | 192 bits | <p>Encrypted authentication key wrapped with KWK. This is a random sequence meant to act as authentication mechanism.</p>                                                                                                                                                                                                                                                                                                                                          |

---

2296 The PKT.SID field is used in this control packet as the Switch where the Service Node is registering. The  
2297 PKT.LNID field is used in this control packet as the Local Node Identifier being assigned to the Service Node  
2298 during the registration process negotiation.

2299 The REG.CAP\_PA field is used to indicate the packet aggregation capability as discussed in Section 4.3.7. In  
2300 the uplink direction, this field is an indication from the registering Terminal Node about its own capabilities.  
2301 For the Downlink response, the Base Node evaluates whether or not all the devices in the cascaded chain  
2302 from itself to this Terminal Node have packet-aggregation capability. If they do, the Base Node shall set  
2303 REG.CAP\_PA=1; otherwise REG.CAP\_PA=0.

MSB

| REG.N     | REG.R | REG.SPC      | REG.CAP_R | REG.CAP_BC  | REG.CAP_SW | REG.CAP_PA      | REG.CAP_CFP | REG.CAP_DC | REG.ALV_F | REG.CAP_MF | REG.CAP_ARQ  | REG.TIME |
|-----------|-------|--------------|-----------|-------------|------------|-----------------|-------------|------------|-----------|------------|--------------|----------|
|           |       | REG.EUI48[0] |           |             |            |                 |             |            |           |            | REG.EUI48[1] |          |
|           |       | REG.EUI48[2] |           |             |            |                 |             |            |           |            | REG.EUI48[3] |          |
|           |       | REG.EUI48[4] |           |             |            |                 |             |            |           |            | REG.EUI48[5] |          |
| REG.PRM_F |       | REG.SAR_SIZE |           | REG.PLC_ATT |            |                 |             |            |           |            | REG.CNT[0]   |          |
|           |       | REG.CNT[1]   |           |             |            |                 |             |            |           |            | REG.CNT[2]   |          |
|           |       | REG.CNT[3]   |           |             |            |                 |             |            |           |            | REG.SWK[0]   |          |
|           |       |              |           |             |            | REG.SWK[1..2]   |             |            |           |            |              |          |
|           |       |              |           |             |            | REG.SWK[3..4]   |             |            |           |            |              |          |
|           |       |              |           |             |            | REG.SWK[5..6]   |             |            |           |            |              |          |
|           |       |              |           |             |            | REG.SWK[7..8]   |             |            |           |            |              |          |
|           |       |              |           |             |            | REG.SWK[9..10]  |             |            |           |            |              |          |
|           |       |              |           |             |            | REG.SWK[11..12] |             |            |           |            |              |          |
|           |       |              |           |             |            | REG.SWK[13..14] |             |            |           |            |              |          |
|           |       |              |           |             |            | REG.SWK[15..16] |             |            |           |            |              |          |
|           |       |              |           |             |            | REG.SWK[17..18] |             |            |           |            |              |          |
|           |       |              |           |             |            | REG.SWK[19..20] |             |            |           |            |              |          |
|           |       |              |           |             |            | REG.SWK[21..22] |             |            |           |            |              |          |
|           |       | REG.SWK[23]  |           |             |            |                 |             |            |           |            | REG.WK[0]    |          |
|           |       |              |           |             |            | REG.WK[1..2]    |             |            |           |            |              |          |
|           |       |              |           |             |            | REG.WK[3..4]    |             |            |           |            |              |          |
|           |       |              |           |             |            | REG.WK[5..6]    |             |            |           |            |              |          |
|           |       |              |           |             |            | REG.WK[7..8]    |             |            |           |            |              |          |
|           |       |              |           |             |            | REG.WK[9..10]   |             |            |           |            |              |          |
|           |       |              |           |             |            | REG.WK[11..12]  |             |            |           |            |              |          |
|           |       |              |           |             |            | REG.WK[13..14]  |             |            |           |            |              |          |
|           |       |              |           |             |            | REG.WK[15..16]  |             |            |           |            |              |          |
|           |       |              |           |             |            | REG.WK[17..18]  |             |            |           |            |              |          |
|           |       |              |           |             |            | REG.WK[19..20]  |             |            |           |            |              |          |
|           |       |              |           |             |            | REG.WK[21..22]  |             |            |           |            |              |          |
|           |       |              |           |             |            |                 |             |            |           |            | REG.WK[23]   |          |

LSB

Figure 58 - REG control packet structure

Table 23 - REG control packet types

| Name      | HDR.DO | PKT.LNID | REG.N | REG.R | Description                                                                                                                                                                                                     |
|-----------|--------|----------|-------|-------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| REG_REQ   | 0      | 0x3FFF   | 0     | R     | Registration request <ul style="list-style-type: none"> <li>If R=0 any previous connection from this Node shall be lost;</li> <li>If R=1 any previous connection from this Node shall be maintained.</li> </ul> |
| REG_RSP   | 1      | < 0x3FFF | 0     | R     | Registration response. This packet assigns the PCK.LNID to the Service Node.                                                                                                                                    |
| REG_ACK   | 0      | < 0x3FFF | 0     | R     | Registration acknowledged by the Service Node.                                                                                                                                                                  |
| REG_REJ   | 1      | 0x3FFF   | 1     | 0     | Registration rejected by the Base Node.                                                                                                                                                                         |
| REG_UNR_S | 0      | < 0x3FFF | 1     | 0     | <ul style="list-style-type: none"> <li>After a REG_UNR_B: Unregistration acknowledge;</li> <li>Alone: Unregistration request initiated by the Node.</li> </ul>                                                  |
| REG_UNR_B | 1      | < 0x3FFF | 1     | 0     | <ul style="list-style-type: none"> <li>After a REG_UNR_S: Unregistration acknowledge;</li> <li>Alone: Unregistration request initiated by the Base Node</li> </ul>                                              |

Fields REG.SWK and REG.WK are of significance only for REG\_RSP messages with Security Profiles 1 and 2 (REG.SCP=1 and REG.SCP=2). For all other message-exchange variants using the REG control packet, these fields shall not be present reducing the length of payload.

In REG\_RSP message, the REG.SWK and REG.WK shall always be inserted wrapped with KWK.

Field REG.CNT is of significance only for REG\_REQ, REG\_RSP and REG\_REJ messages with Security Profiles 1 and 2 (REG.SCP=1 and REG.SCP=2). For all other message-exchange variants using the REG control packet, these fields shall not be present reducing the length of payload.

#### 4.4.2.6.4 CON control packet (PKT.CTYPE = 2)

This control packet is used for negotiating the connections. The description of the fields of this packet is given in Table 24 and Figure 59 The meaning of the packet differs depending on the direction of the packet and on the values of the different types.

Table 25 shows the different interpretation of the packets. The packets are used during the connection establishment and closing.

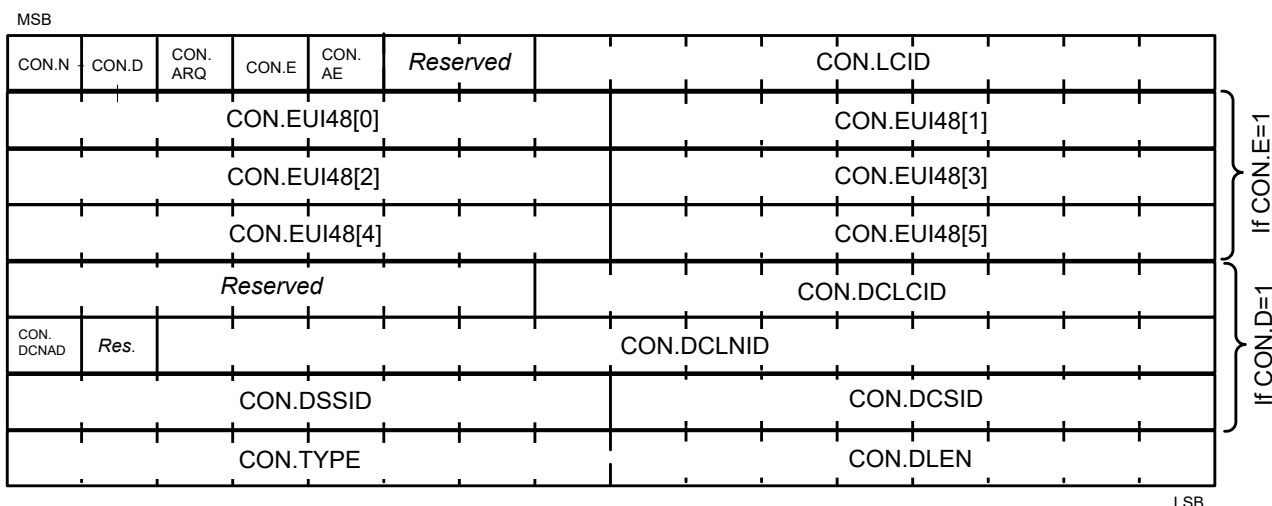


Figure 59 - CON control packet structure

Note that Figure 59 shows the complete message with all optional parts. When CON.D is 0, CON.DCNAD, CON.DSSID, CON.DCLNID, CON.DCLCID, CON.DCSID and the reserved field between CON.DCNAD and CON.DSSID shall not be present in the message. Thus, the message shall be 6 octets smaller. Similarly, when CON.E is zero, the field CON.EUI-48 shall not be present, making the message 6 octets smaller.

Table 24 - CON control packet fields

| Name    | Length | Description                                                                                                                                                                                                                          |
|---------|--------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| CON.N   | 1 bit  | Negative <ul style="list-style-type: none"> <li>CON.N=1 for the negative connection;</li> <li>CON.N=0 for the positive connection.</li> </ul>                                                                                        |
| CON.D   | 1 bit  | Direct connection <ul style="list-style-type: none"> <li>CON.D=1 if information about direct connection is carried by this packet;</li> <li>CON.D=0 if information about direct connection is not carried by this packet.</li> </ul> |
| CON.ARQ | 1 bit  | ARQ mechanism enable <ul style="list-style-type: none"> <li>CON.ARQ=1 if ARQ mechanism is enabled for this connection;</li> <li>CON.ARQ=0 if ARQ mechanism is not enabled for this connection.</li> </ul>                            |
| CON.E   | 1 bit  | EUI-48 presence <ul style="list-style-type: none"> <li>CON.E = 1 to have a CON.EUI-48;</li> <li>CON.E = 0 to not have a CON.EUI-48 so that this connection establishment is for reaching the Base Node CL.</li> </ul>                |

| Name            | Length                          | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             |
|-----------------|---------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| CON.AE          | 1 bit                           | <p>Use authentication and encryption on the data sent using this connection.</p> <ul style="list-style-type: none"> <li>CON.AE = 1 to encrypt and authenticate the data packets.</li> <li>CON.AE = 0 to send the plain data without any authentication and encryption.</li> </ul> <p>This flag is valid in both directions; each side shall set it to the desired value. The highest security will be applied in case the values do not match. So if any side sets this flag to 1 the data shall be authenticated and encrypted.</p> <p>NOTE: This flag only can be 1 on security profiles 1 and 2.</p> |
| <i>Reserved</i> | 2 bits                          | <p>Reserved for future version of the protocol.</p> <p>This shall be 0 for this version of the protocol.</p>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            |
| CON.LCID        | 9 bits                          | <p>Local Connection Identifier.</p> <p>The LCID is reserved in the connection request. LCIDs from 0 to 255 are assigned by the connection requests initiated by the Base Node. LCIDs from 256 to 511 are assigned by the connection requests initiated by the local Node.</p> <p>This is the identifier of the connection being managed with this packet. This is not the same as the PKT.LCID of the generic header, which does not exist for control packets.</p>                                                                                                                                     |
| CON.EUI-48      | 48 bits<br>(Present if CON.E=1) | <p>EUI-48 of destination/source Service Node/Base Node for connection request.</p> <p>When not performing a directed connection, this field shall not be included. When performing a directed connection, it may contain the SNA, indicating that the Base Node Convergence layer shall determine the EUI-48.</p> <ul style="list-style-type: none"> <li>CON.D = 0, Destination EUI-48;</li> <li>CON.D = 1, Source EUI-48.</li> </ul>                                                                                                                                                                   |
| <i>Reserved</i> | 7 bits<br>(Present if CON.D=1)  | <p>Reserved for future version of the protocol.</p> <p>This shall be 0 for this version of the protocol.</p>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            |
| CON.DCLCID      | 9 bits<br>(Present if CON.D=1)  | <p>Direct Connection LCID</p> <p>This field represents the LCID of the connection identifier to which the one being established shall be directly switched.</p>                                                                                                                                                                                                                                                                                                                                                                                                                                         |

| Name            | Length                                | Description                                                                                                                                                                                                                                                                                           |
|-----------------|---------------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| CON.DCNAD       | 1 bit<br>(Present if CON.D=1)         | Reserved for future version of the protocol. Direct Connection Not Aggregated at Destination<br>This field represents the content of the PKT.NAD field after a direct connection Switch operation.                                                                                                    |
| <i>Reserved</i> | 1 bits<br>(Present if CON.D=1)        | Reserved for future version of the protocol.<br>This shall be 0 for this version of the protocol.                                                                                                                                                                                                     |
| CON.DCLNID      | 14 bits<br>(Present if CON.D=1)       | Direct Connection LNID<br>This field represents the LNID part of the connection identifier to which the one being established shall be directly switched.                                                                                                                                             |
| CON.DSSID       | 8 bits<br>(Present if CON.D=1)        | Direct Switch SID<br>This field represents the SID of the Switch that shall learn this direct connection and perform direct switching.                                                                                                                                                                |
| CON.DCSID       | 8 bits<br>(Present if CON.D=1)        | Direct Connection SID<br>This field represents the SID part of the connection identifier to which the one being established shall be directly switched.                                                                                                                                               |
| CON.TYPE        | 8 bits                                | Connection type.<br>The connection type (see Annex E) specifies the Convergence layer to be used for this connection. They are treated transparently through the MAC common part sublayer, and are used only to identify which Convergence layer may be used.                                         |
| CON.DLEN        | 8 bits                                | Length of CON.DATA field in bytes                                                                                                                                                                                                                                                                     |
| CON.DATA        | (variable)<br>(Present if CON.DLEN>0) | Connection specific parameters.<br>These connections specific parameters are Convergence layer specific. They shall be defined in each Convergence layer to define the parameters that are specific to the connection. These parameters are handled in a transparent way by the common part sublayer. |

2330

2331

Table 25 - CON control packet types

| Name      | HDR.DO | CON.N | Description                                                     |
|-----------|--------|-------|-----------------------------------------------------------------|
| CON_REQ_S | 0      | 0     | Connection establishment request initiated by the Service Node. |

| Name      | HDR.DO | CON.N | Description                                                                                                                                                                                                                                                                                                   |
|-----------|--------|-------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| CON_REQ_B | 1      | 0     | The Base Node shall consider that the connection is established with the identifier CON.LCID. <ul style="list-style-type: none"><li>• After a CON_REQ_S: Connection accepted;</li><li>• Alone: Connection establishment request.</li></ul>                                                                    |
| CON_CLS_S | 0      | 1     | The Service Node considers this connection closed: <ul style="list-style-type: none"><li>• After a CON_REQ_B: Connection rejected by the Node;</li><li>• After a CON_CLS_B: Connection closing acknowledge;</li><li>• Alone: Connection closing request.</li></ul>                                            |
| CON_CLS_B | 1      | 1     | The Base Node shall consider that the connection is no longer established. <ul style="list-style-type: none"><li>• After a CON_REQ_S: Connection establishment rejected by the Base Node;</li><li>• After a CON_CLS_S: Connection closing acknowledge;</li><li>• Alone: Connection closing request.</li></ul> |

4.4.2.6.5 PRO control packet (PKT.CTYPE = 3)

This control packet is used to promote a Service Node from Terminal function to Switch function. This control packet is also used to exchange information that is further used by the Switch Node to transmit its beacon. The description of the fields of this packet is given in Table 26, and Figure 61. The meaning of the packet differs depending on the direction of the packet and on the values of the different types.

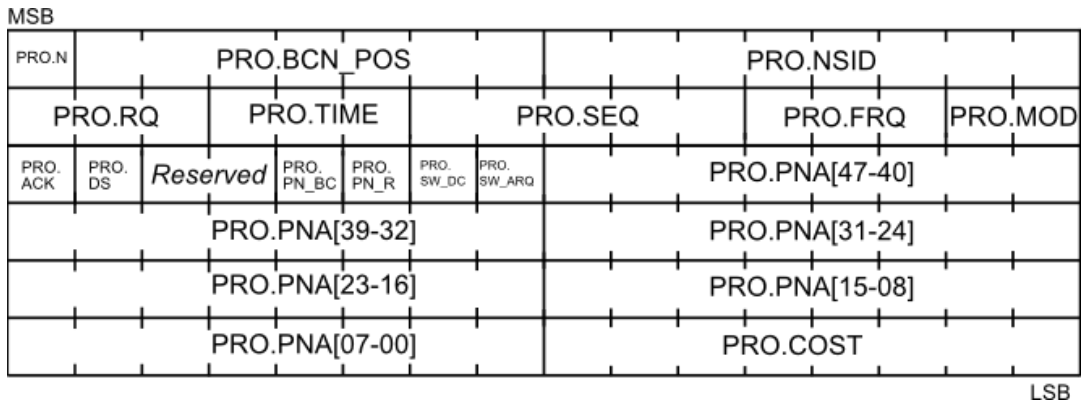


Figure 60 - PRO\_REQ\_S control packet structure

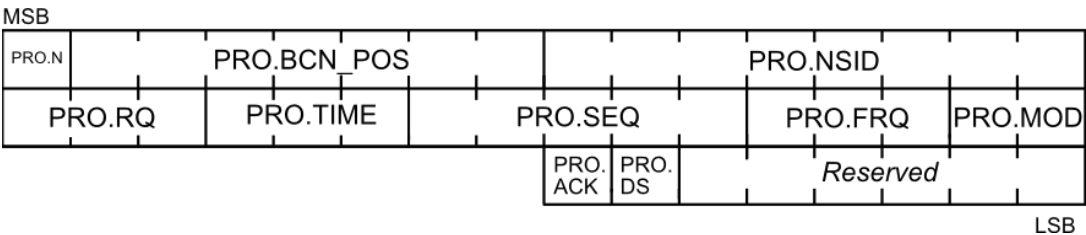


Figure 61 - PRO control packet structure

Note that Figure 60 includes all fields as used by a PRO\_REQ\_S message. All other messages are much smaller, containing only fields shown in Figure 61.



Table 26 - PRO control packet fields

| Name        | Length | Description                                                                                                                                                                                                                                                                                                                                      |
|-------------|--------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| PRO.N       | 1 bit  | Negative<br>PRO.N=1 for the negative promotion<br>PRO.N=0 for the positive promotion                                                                                                                                                                                                                                                             |
| PRO.BCN_POS | 7 bits | Position of this beacon in symbols from the beginning of the frame.                                                                                                                                                                                                                                                                              |
| PRO.NSID    | 8 bits | New Switch Identifier.<br><br>This is the assigned Switch identifier of the Node whose promotion is being managed with this packet. This is not the same as the PKT.SID of the packet header, which must be the SID of the Switch this Node is connected to, as a Terminal Node.                                                                 |
| PRO.RQ      | 3 bits | Receive quality of the PNPDU message received from the Service Node requesting the Terminal to promote.                                                                                                                                                                                                                                          |
| PRO.TIME    | 3 bits | The ALV.TIME that is being used by the terminal that shall become a switch. On a reception of this time in a PRO_REQ_B the Service Node shall reset the Keep-Alive timer in the same way as receiving an ALV_REQ_B.                                                                                                                              |
| PRO.SEQ     | 5 bits | The Beacon Sequence number when the specified change takes effect.                                                                                                                                                                                                                                                                               |
| PRO.FRQ     | 3 bits | Transmission frequency of Beacon, encoded as:<br><br>FRQ = 0 => 1 beacon every frame<br>FRQ = 1 => 1 beacon every 2 frames<br>FRQ = 2 => 1 beacon every 4 frames<br>FRQ = 3 => 1 beacon every 8 frames<br>FRQ = 4 => 1 beacon every 16 frames<br>FRQ = 5 => 1 beacon every 32 frames<br>FRQ = 6 => <i>Reserved</i><br>FRQ = 7 => <i>Reserved</i> |
| PRO.MOD     | 2 bits | For PLC, the modulation of the transmitted Beacons, is encoded as:<br><br>ENC = 0 => DBPSK + Convolutional Code<br>ENC = 1 => Robust DQPSK<br>ENC = 2 => Robust DBPSK<br>ENC = 3 => <i>Reserved</i><br><br>For RF, this field is reserved for future use.                                                                                        |
| PRO.ACK     | 1 bit  | Flag to differentiate the PRO_REQ_S from the PRO_ACK                                                                                                                                                                                                                                                                                             |
| PRO.DS      | 1 bit  | Double switch flag. Used for switches that have to send a second beacon. This field is described in more detail in section 4.6.3.                                                                                                                                                                                                                |

|                 |              |                                                                                                                                                                                                                                                     |
|-----------------|--------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>Reserved</i> | 2 bits       | Reserved for future versions of the protocol. Shall be set to 0 for this version of the protocol.                                                                                                                                                   |
| PRO.PN_BC       | 1 bit        | Backwards Compatibility mode of the node represented by PRO.PNA.<br>1 if the device is backwards compatible with 1.3.6 PRIME<br>0 if it is not                                                                                                      |
| PRO.PN_R        | 1 bit        | Robust mode compatibility of the node represented by PRO.PNA.<br>1 if the device supports robust mode<br>0 if it is not                                                                                                                             |
| PRO.SWC_DC      | 1 bit        | Direct Connection Switching Capability<br>1 if the device is able to behave as Direct Switch in direct connections.<br>0 otherwise                                                                                                                  |
| PRO.SWC_ARQ     | 1 bit        | ARQ Buffering Switching Capability<br>1 if the device is able to perform buffering for ARQ connections while switching.<br>0 if the device is not able to perform buffering for ARQ connections while switching.                                    |
| PRO.PNA         | 0 or 48 bits | Promotion Need Address, contains the EUI-48 of the Terminal requesting the Service Node promotes to become a Switch.<br>This field is only included in the PRO_REQ_S message.                                                                       |
| PRO.COST        | 0 or 8 bits  | Total cost from the Terminal Node to the Base Node. This value is calculated in the same way a Switch Node calculates the value it places into its own Beacon PDU.<br>This field is only included in the PRO_REQ_S and PRO_REQ_S_MultiPHY messages. |

Table 27 - PRO control packet types

| Name      | HDR.<br>DO | PRO.<br>N | PRO.<br>ACK | PRO.<br>NSID | Description                                                                                                                                                                                                                                                                                                                                            |
|-----------|------------|-----------|-------------|--------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| PRO_REQ_S | 0          | 0         | 0           | -            | Used by terminal nodes to request a promotion to switch nodes. This is not part of any procedure, just an information message, so there shall not be any PRO_ACK/PRO_NACK in response.<br><br>Used by switch nodes to request a beacon modulation change. In this case it is a procedure, so the base node shall respond with a PRO_ACK/PRO_NACK.      |
| PRO_REQ_B | 1          | 0         | 0           | < 0xFF       | The Base Node shall consider that the Service Node has promoted with the identifier PRO.NSID.<br><br><ul style="list-style-type: none"> <li>To a terminal node: Promotion acceptance with allocating LSID or Promotion request initiated by the Base Node.</li> <li>To a switch node: Beacon information change initiated by the Base Node.</li> </ul> |
| PRO_ACK   | -          | 0         | 1           | < 0xFF       | Acknowledge. Used by both the Base Node and the Service Node to acknowledge with a positive answer the procedure.<br><br>Procedures this message applies to are: PRO_REQ_S for beacon change modulation or PRO_REQ_B.                                                                                                                                  |
| PRO_NACK  | -          | 1         | 1           | < 0xFF       | Negative Acknowledge. Used by both the Base Node and the Service Node to acknowledge with a negative answer the procedure.<br><br>Procedures this message applies to be: beacon change modulation.                                                                                                                                                     |
| PRO_DEM_S | 0          | 1         | 0           | < 0xFF       | Used by Service Nodes to request a demotion, to reject a promotion or to positively acknowledge a demotion.                                                                                                                                                                                                                                            |
| PRO_DEM_B | 1          | 1         | 0           | -            | Used by the Base Node to request a demotion, to reject a promotion or to positively acknowledge a demotion.                                                                                                                                                                                                                                            |

2346

2347 Table 27 shows the different interpretation of the packets. The promotion process is explained in more detail  
 2348 in 4.6.3.

#### 2349 4.4.2.6.5.1 Extension for Multi-PHY promotion

2350 To support the extension to handle Multiple PHY layers, the reserved bytes of the PRO\_REQ\_S and  
 2351 PRO\_REQ\_B messages are used to create new versions of the messages named PRO\_REQ\_S\_MultiPHY  
 2352 (Figure 62) and PRO\_REQ\_B\_MultiPHY (Figure 63), where some not used fields have been also redefined.

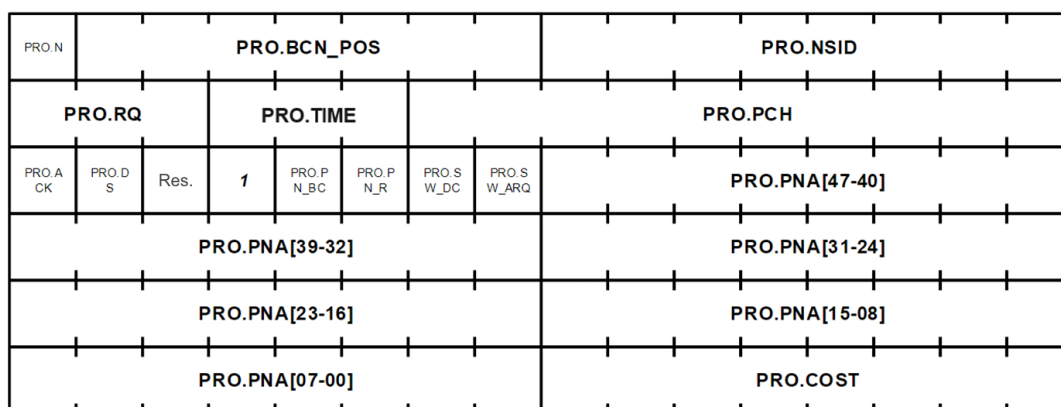
2353 The new messages and their use are defined in Table 28 and the new fields in Table 29

2354

Table 28 - Extension for Multi PHY Promotion

| Name               | HDR.<br>DO | PRO.<br>N | PRO.<br>ACK | Reserve<br>d | Description                                                                                                                                                                                                  |
|--------------------|------------|-----------|-------------|--------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| PRO_REQ_S_MultiPHY | 0          | 0         | 0           | 01b          | Used by service nodes to request a promotion in a different physical medium from the one where the node is connected to the network. This physical medium could be either RF or PLC                          |
| PRO_REQ_B_MultiPHY | 1          | 0         | 0           | 100xxxb      | Used by the Base Node to accept a PRO_REQ_S_MultiPHY promotion request from a service node or to request a promotion in a different physical medium from the one where the node is connected to the network. |

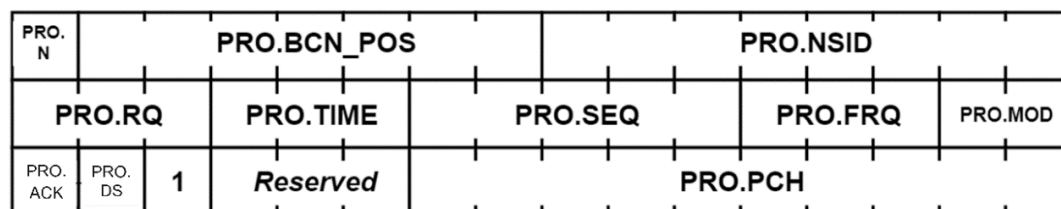
2355



2356

2357

Figure 62 - PRO\_REQ\_S Multi PHY control packet structure



2358

2359

Figure 63- PRO\_REQ\_B Multi PHY control packet structure

2360

2361

2362

2363

2364

2365

2366

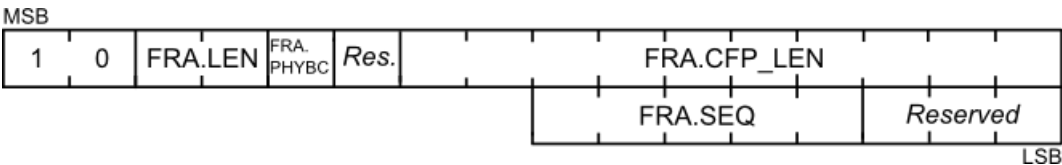
Table 29 – New fields defined in the MultiPHY PRO messages

| Name    | Length  | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           |
|---------|---------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| PRO.PCH | 10 bits | <p>Codification of the Physical layer to where a new switch is requested.</p> <ul style="list-style-type: none"><li>– PRO.PCH[09]=0b for the PLC PHY</li><li>– PRO.PCH[09]=1b for the SUN FSK PHY, based on clause 20 of [28,29]</li></ul> <p>If PRO.PCH[9]=0 for the PLC PHY</p> <ul style="list-style-type: none"><li>– PRO.PCH[08]=0b reserved</li><li>– PRO.PCH[07-00] PLC channels used. As examples;<ul style="list-style-type: none"><li>– PRO.PCH[07-00]=01H if Channel 1 only is used .</li><li>– PRO.PCH[07-00]=80H if Channel 8 only is used .</li><li>– PRO.PCH[07-00]=FCH if Channel 3 to 8 are used.</li></ul></li></ul> <p>If PRO.PCH[9]=1 for the RF PHY</p> <ul style="list-style-type: none"><li>– PRO.PCH[08-00] RF channels used, <i>NumChan</i>, according to channel numbering defined on [28,29] (10.1.2.8) from 0 to TotalNumChan-1</li></ul> |

2367

2368 **4.4.2.6.6 FRA control packet (PKT.CTYPE = 5)**

2369 This control packet is broadcast from the Base Node and relayed by all Switch Nodes to the entire  
2370 Subnetwork. It is used to circulate information on the change of Frame structure at a specific time in future.  
2371 The description of fields of this packet is given inTable 30 and Figure 64.



2372

2373

Figure 64 - FRA control packet structure

2374

Table 30 - FRA control packet fields

| Name            | Length | Description                                                                                                                                                          |
|-----------------|--------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>Reserved</i> | 2 bits | Reserved bits. Shall be set to 0b10 for this version of this specification                                                                                           |
| FRA.LEN         | 2 bits | <p>Length of the frame to be applied in the next superframe. This shall be the <i>macFrameLength</i>, encoded with same semantics as the PIB attribute.</p> <p>•</p> |

| Name                          | Length  | Description                                                                                                                                                                                                                                                                                                                                             |
|-------------------------------|---------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>FRA.PHYB</i><br><i>C</i>   | 1 bit   | The network is working on PHY backwards compatibility mode, all the nodes that need to send Type B PHY Frames shall use PHY backwards compatible frames. <ul style="list-style-type: none"> <li>0 if the subnet is not working in PHY backwards compatibility mode.</li> <li>1 if the subnet is working in PHY backwards compatibility mode.</li> </ul> |
| <i>Reserved</i>               | 1 bit   | Reserved for future version of this protocol. In this version, this field shall be initialized to 0.                                                                                                                                                                                                                                                    |
| <i>FRA.CFP_</i><br><i>LEN</i> | 10 bits | Length of CFP.                                                                                                                                                                                                                                                                                                                                          |
| <i>FRA.SEQ</i>                | 5 bits  | The Beacon Sequence number when the specified change takes effect.                                                                                                                                                                                                                                                                                      |
| <i>Reserved</i>               | 3 bits  | Reserved for future version of this protocol. In this version this field shall be set to 0.                                                                                                                                                                                                                                                             |

#### 4.4.2.6.7 CFP control packet (PKT.CTYPE = 6)

This control packet is used for dedicated contention-free channel access time allocation to individual Terminal or Switch Nodes. The description of the fields of this packet is given in

Table 31 and Figure 65. The meaning of the packet differs depending on the direction of the packet and on the values of the different types.

Table 32 represents the different interpretation of the packets.

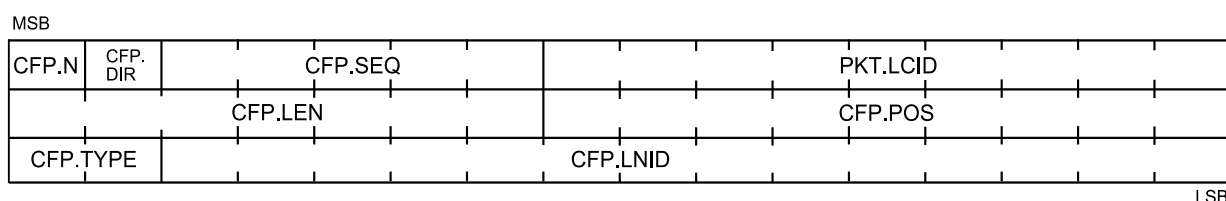


Figure 65 - CFP control packet structure

Table 31 - CFP control message fields

| Name     | Length | Description                                                                                                                                                                                |
|----------|--------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| CFP.N    | 1 bit  | 0: denial of allocation/deallocation request;<br>1: acceptance of allocation/deallocation request.                                                                                         |
| CFP.DIR  | 1 bit  | Indicate direction of allocation.<br><br>0: allocation is applicable to uplink (towards Base Node) direction;<br>1: allocation is applicable to Downlink (towards Service Node) direction. |
| CFP.SEQ  | 5 bits | The Beacon Sequence number when the specified change takes effect.                                                                                                                         |
| CFP.LCID | 9 bits | LCID of requesting connection.                                                                                                                                                             |

| Name     | Length  | Description                                                                                    |
|----------|---------|------------------------------------------------------------------------------------------------|
| CFP.LEN  | 7 bits  | Length (in symbols) of requested/allocated channel time per frame.                             |
| CFP.POS  | 9 bits  | Offset (in symbols) of allocated time from beginning of frame.                                 |
| CFP.TYPE | 2 bits  | 0: Channel allocation packet;<br>1: Channel de-allocation packet;<br>2: Channel change packet. |
| CFP.LNID | 14 bits | LNID of Service Node that is the intended user of the allocation.                              |

2386

2387

Table 32 - CFP control packet types

| Name          | CFP.TYP | HDR.DO | Description                                                                                                                                                      |
|---------------|---------|--------|------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| CFP_ALC_REQ_S | 0       | 0      | Service Node makes channel allocation request                                                                                                                    |
| CFP_ALC_IND   | 0       | 1      | <ul style="list-style-type: none"><li>After a CFP_ALC_REQ_S: Requested channel is allocated</li><li>Alone: Unsolicited channel allocation by Base Node</li></ul> |
| CFP_ALC_REJ   | 0       | 1      | Requested channel allocation is denied                                                                                                                           |
| CFP_DALC_REQ  | 1       | 0      | Service Node makes channel de-allocation request                                                                                                                 |
| CFP_DALC_RSP  | 1       | 1      | Base Node confirms de-allocation                                                                                                                                 |
| CFP_CHG_IND   | 2       | 1      | Change of location of allocated channel within the CFP.                                                                                                          |

2388

4.4.2.6.8 ALV control packet (PKT.CTYPE = 7)

2389

The ALV control message is used for Keep-Alive signaling between a Service Node, the Service Nodes above it and the Base Node. It is also used to test every hop in the path of that particular node performing robustness-management. Structures of these messages are shown in Figure 66, Figure 67 and Figure 68 and individual fields are enumerated in

2390

2391

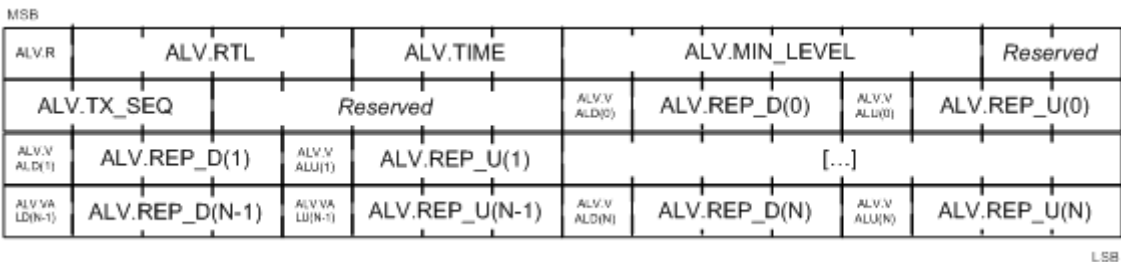
2392

2393

Table 33. The different Keep-Alive message types are shown in ALV.VALU(\*) to zero.

2394

Table 34. These messages are sent periodically, as described in section 4.6.5.



2395

2396

Figure 66 - ALV\_RSP\_S / ALV\_REQ\_B Control packet structure

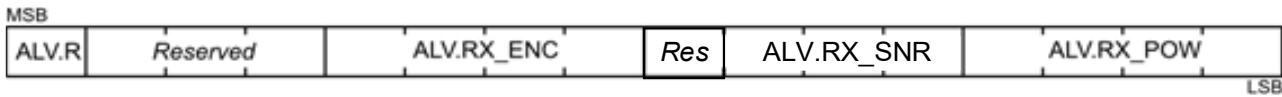


Figure 67 - ALV\_ACK\_B/ALV\_ACK\_S control packet structure

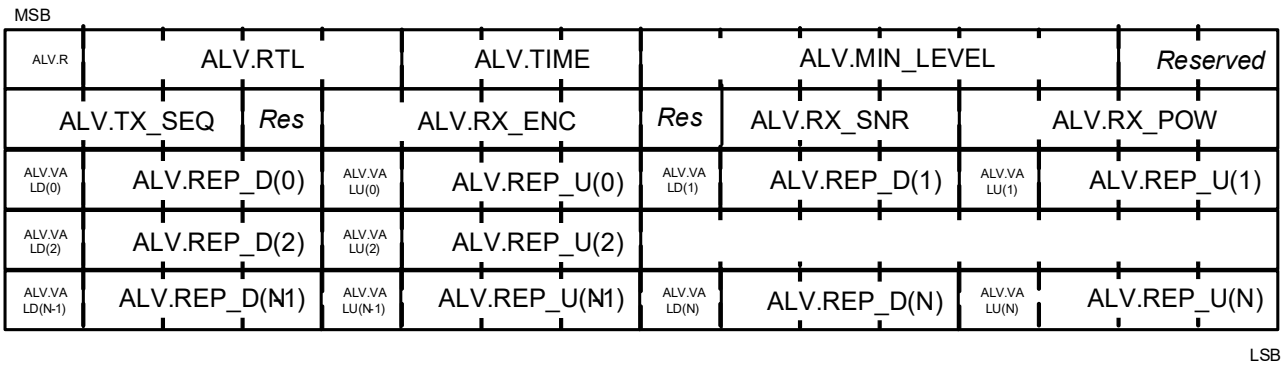


Figure 68 - ALV\_RSP\_ACK Control packet structure

Table 33 - ALV control message fields



| Name          | Length | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |
|---------------|--------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| ALV.R         | 1 bit  | Request/Response field. <ul style="list-style-type: none"> <li>1 in the requests/response</li> <li>0 in the acknowledges</li> </ul>                                                                                                                                                                                                                                                                                                                                                                                                                |
| ALV.RTL       | 4 bits | Total number of repetitions left across the entire path.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           |
| ALV.TIME      | 3 bits | Time to wait for an ALV procedure before assuming the Service Node has been unregistered by the Base Node. <p>ALV.TIME = 0 =&gt; 128 seconds ~ 2.1 minutes;<br/> ALV.TIME = 1 =&gt; 256 seconds ~ 4.2 minutes;<br/> ALV.TIME = 2 =&gt; 512 seconds ~ 8.5 minutes;<br/> ALV.TIME = 3 =&gt; 2048 seconds ~ 34.1 minutes;<br/> ALV.TIME = 4 =&gt; 4096 seconds ~ 68.3 minutes;<br/> ALV.TIME = 5 =&gt; 8192 seconds ~ 136.5 minutes;<br/> ALV.TIME = 6 =&gt; 16384 seconds ~ 273.1 minutes;<br/> ALV.TIME = 7 =&gt; 32768 seconds ~ 546.1 minutes</p> |
| ALV.MIN_LEVEL | 6 bits | Minimum level of the node to add independent records to the REP_D/REP_U table. For downlink case, if the switch's level+1 is equal or lower than ALV.MIN_LEVEL it shall sum its repetitions to the first record (record number 0: ALV.REP_U(0)). For uplink case, if the switch's (or terminal's) level is equal or lower than ALV.MIN_LEVEL it shall sum its repetitions to the first record.                                                                                                                                                     |
| Reserved      | 2 bit  | Reserved for future use. Shall be 0 for this version of the specification.                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |
| ALV.TX_SEQ    | 3 bits | Sequence of number of transmissions, to keep track if the loss in the ALV process is due to the REQ/RSP process or the ACK. This is to avoid an incorrect evaluation of downlink/uplink because of ACK loses. All the ALV operations shall start with sequence number 0, every time a node starts a hop level operation it shall set this field to 0, and each repetition it shall increase it until ACK is received.                                                                                                                              |
| ALV.VALD(*)   | 1 bit  | Flag to indicate that the REP_D record contains valid information. <ul style="list-style-type: none"> <li>1 information contained in REP_D records is valid</li> <li>0 information in REP_D shall be discarded</li> </ul>                                                                                                                                                                                                                                                                                                                          |
| ALV.REP_D(*)  | 3 bits | Number of repetitions for the given downlink hop. Valid values: <ul style="list-style-type: none"> <li>0-5 : Number of repetitions</li> <li>6 : 6 or more repetitions (for record as a sum of various levels)</li> <li>7 : All the retries finished for this hop</li> </ul>                                                                                                                                                                                                                                                                        |
| ALV.VALU(*)   | 1 bit  | Flag to indicate that the REP_U record contains valid information. <ul style="list-style-type: none"> <li>1 information contained in REP_D records is valid</li> <li>0 information in REP_D shall be discarded</li> </ul>                                                                                                                                                                                                                                                                                                                          |

|              |        |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 |
|--------------|--------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| ALV.REP_U(*) | 3 bits | <p>Number of repetitions for the given uplink hop.</p> <p>In the ALV_REQ_B the base node shall fill these fields with the repetitions of each hop for the last ALV procedure of that hop.</p> <p>In the ALV_RSP_S the service node shall fill the field with the uplink repetitions.</p> <p>Valid values:</p> <ul style="list-style-type: none"> <li>• 0-5 : Number of repetitions</li> <li>• 6 : 6 or more repetitions (for record as a sum of various levels)</li> <li>• 7 : All the retries finished for this hop</li> </ul> |
| ALV.RX_SNR   | 3 bits | Signal to Noise Ratio at which the ALV_REQ_B/ALV_RSP_S was received. It corresponds to the SNR parameter defined in 3.5.3.12.2 (values from 0 to 7).                                                                                                                                                                                                                                                                                                                                                                            |
| ALV.RX_POW   | 4 bits | Power at which the ALV_REQ_B/ALV_RSP_S was received. It corresponds to the Level parameter defined in 3.5.2.4.2.                                                                                                                                                                                                                                                                                                                                                                                                                |
| ALV.RX_ENC   | 4 bits | <p>Encoding at which the ALV_REQ_B/ALV_RSP_S was received.</p> <ul style="list-style-type: none"> <li>• 0 – DBPSK</li> <li>• 1 – DQPSK</li> <li>• 2 – D8PSK</li> <li>• 3 – Not used</li> <li>• 4 – DBPSK + Convolutional Code</li> <li>• 5 – DQPSK + Convolutional Code</li> <li>• 6 – D8PSK + Convolutional Code</li> <li>• 7-11 – Not used</li> <li>• 12 – Robust DBPSK</li> <li>• 13 – Robust DQPSK</li> <li>• 14 – Not used</li> <li>• 15 – Outdated information</li> </ul>                                                 |

2402

2403 The \* symbol means that there are a variable number of records for the same field, each record shall be  
 2404 fulfilled by a Service Node in the path to the Terminal Node that shall receive the ALV, Position N shall be  
 2405 the hop of the Service Node target of the ALV procedure, N-1 shall be the parent Switch Node of that node,  
 2406 and so on. For the switches with level below or equal than ALV.MIN\_LEVEL shall add their repetitions  
 2407 information to the record ALV.REP\_U(0)/ALV.REP\_D(0). The Base Node shall make sure that the number of  
 2408 records is correct for the given ALV.MIN\_LEVEL value.

2409 The base node shall fill the ALV.REP\_U(\*) registries with the last ALV operation's uplink retries for each hop,  
 2410 if it does not have that information it shall reset the appropriate ALV.VALU(\*) to zero.

2411

Table 34 – Keep-Alive control packet types

| Name        | HDR.DO | ALV.R | Description                                                                                            |
|-------------|--------|-------|--------------------------------------------------------------------------------------------------------|
| ALV_REQ_B   | 1      | 1     | Keep-Alive request message.                                                                            |
| ALV_ACK_B   | 1      | 0     | Keep-Alive acknowledge to a response.                                                                  |
| ALV_RSP_S   | 0      | 1     | Keep-Alive response message in case the node is not the target node (PKT.SID != receiver SSID)         |
| ALV_RSP_ACK | 0      | 1     | Keep-Alive response acknowledge message in case the node is the target node (PKT.SID == receiver SSID) |
| ALV_ACK_S   | 0      | 0     | Keep-Alive acknowledge to a request.                                                                   |

4.4.2.6.9 MUL control packet (PKT.CTYPE = 8)

The MUL message is used to control multicast group membership. The structure of this message and the meanings of the fields are described in Table 35 and Figure 69. The message can be used in different ways as described in

Table 36.

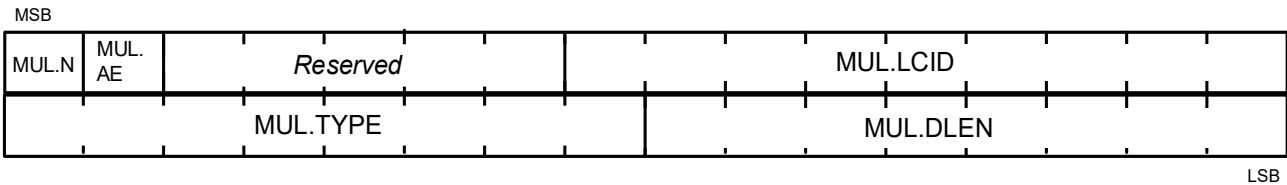


Figure 69 - MUL control packet structure

Table 35 - MUL control message fields

| Name  | Length | Description                                                                                                                                                                                                               |
|-------|--------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| MUL.N | 1 bit  | Negative <ul style="list-style-type: none"><li>MUL.N = 1 for the negative multicast connection, i.e. multicast group leave.</li><li>MUL.N = 0 for the positive multicast connection, i.e. multicast group join.</li></ul> |

| Name     | Length                                       | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |
|----------|----------------------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| MUL.AE   | 1 bit                                        | <p>Use authentication and encryption on the data sent using this multicast connection.</p> <ul style="list-style-type: none"> <li>MUL.AE = 1 to encrypt and authenticate the data packets.</li> <li>MUL.AE = 0 to send the plain data without any authentication and encryption.</li> </ul> <p>This flag is valid in both directions; each side shall set it to the desired value. The highest security will be applied in case the values do not match. So if any side sets this flag to 1 the data shall be authenticated and encrypted.</p> <p><b>NOTE:</b> This flag only can be 1 on security profiles 1 and 2.</p> |
| Reserved | 5 bits                                       | Reserved for future version of the protocol. This shall be 0 for this version of the protocol.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           |
| MUL.LCID | 9 bits                                       | Local Connection Identifier. The LCID indicates which multicast distribution group is being managed with this message.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |
| MUL.TYPE | 8 bits                                       | Connection type. The connection type specifies the Convergence layer to be used for this connection. They are treated transparently through the MAC common part sublayer, and are used only to identify which Convergence layer may be used. See Annex E.                                                                                                                                                                                                                                                                                                                                                                |
| MUL.DLEN | 8 bits                                       | Length of data in bytes in the MUL.DATA field                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            |
| MUL.DATA | (variable)(<br>Present if<br>MUL.DLEN<br>>0) | Connection specific parameters. These connections specific parameters are Convergence layer specific. They shall be defined in each Convergence layer to define the parameters that are specific to the connection. These parameters are handled in a transparent way by the common part sublayer.                                                                                                                                                                                                                                                                                                                       |

2421

2422

Table 36 – MUL control message types

| Name       | HDR.DO | MUL.N | PKT.LNID | Description                                                                                                                                                                                                               |
|------------|--------|-------|----------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| MUL_JOIN_S | 0      | 0     | >0       | Multicast group join request initiated by the Service Node, or an acknowledgement when sent in response to a MUL_JOIN_B.                                                                                                  |
| MUL_JOIN_B | 1      | 0     | >0       | <p>The Base Node shall consider that the group has been joined with the identifier MUL.LCID.</p> <ul style="list-style-type: none"> <li>After a MUL_JOIN_S: join accepted;</li> <li>Alone: group join request.</li> </ul> |

| Name           | HDR.DO | MUL.N | PKT.LNID | Description                                                                                                                                                                                                                                                                                                           |
|----------------|--------|-------|----------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| MUL_LEAVE_S    | 0      | 1     | >0       | <p>The Service Node leaves the multicast group:</p> <ul style="list-style-type: none"> <li>• After a MUL_JOIN_B: Join rejected by the Node;</li> <li>• After a MUL_LEAVE_B: group leave acknowledge;</li> <li>• Alone: group leave request.</li> </ul>                                                                |
| MUL_LEAVE_B    | 1      | 1     | >0       | <p>The Base Node shall consider that the Service Node is no longer a member of the multicast group.</p> <ul style="list-style-type: none"> <li>• After a MUL_JOIN_S: Group join rejected by the Base Node;</li> <li>• After a MUL_LEAVE_S: Group leave acknowledge;</li> <li>• Alone: Group leave request.</li> </ul> |
| MUL_SW_LEAVE_B | 1      | 1     | 0        | <p>The switch node shall stop switching multicast data for the multicast group.</p> <p>This message is always initiated by the base node.</p> <p>The addressing shall be with the switch's SSID and LNID == 0 to distinguish this message from MUL_LEAVE_B.</p>                                                       |
| MUL_SW_LEAVE_S | 0      | 1     | 0        | <p>The switch node is no longer switching multicast data for the multicast group.</p> <p>This message is sent as a response to MUL_SW_LEAVE_B.</p> <p>The addressing shall be with the switch's SSID and LNID == 0 to distinguish this message from MUL_LEAVE_S.</p>                                                  |

#### 2423 4.4.2.6.10 SEC control packet (PKT.CTYPE = 10)

2424 The SEC control message is a unicast message transmitted authenticated and encrypted (WK) by the Base  
 2425 Node to every node in the Subnetwork to update the WK and SWK. The random sequence used by devices  
 2426 in a Subnetwork is dynamic and changes from time to time to ensure a robust security framework. The  
 2427 structure of this message is shown in Table 37 and Figure 70. Further details of security mechanisms are given  
 2428 in Section 4.3.8.

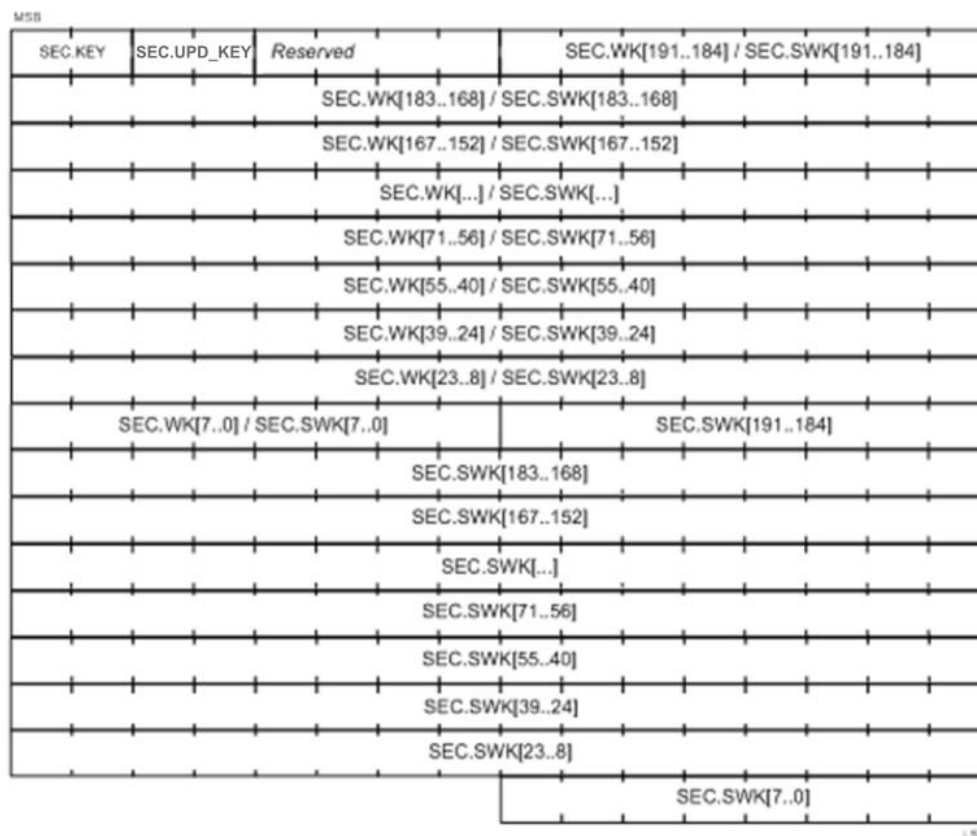


Figure 70 – SEC control packet structure

Table 37 – SEC control message fields

| Name            | Length   | Description                                                                                                                                                                                                                                                                                                                |
|-----------------|----------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| SEC.KEY         | 2 bits   | In the SEC_REQ it indicates which key is present<br>In the SEC_RSP it indicates which key was to be updated<br>0 - reserved<br>1 – only SEC.WK is present / only SEC.WK to be updated<br>2 – only SEC.SWK is present / only SEC.SWK to be updated<br>3 – SEC.WK and SEC.SWK are present / SEC.WK and SEC.SWK to be updated |
| SEC.UPDATED_KEY | 2 bits   | In the SEC_RSP it indicates which key has being updated<br>0 - reserved<br>1 – only SEC.WK was updated<br>2 – only SEC.SWK was updated<br>3 – SEC.WK and SEC.SWK were updated<br>Only used in SEC_RSP. In all other message variants, this shall be 0.                                                                     |
| Reserved        | 4 bits   | Shall always be encoded as 0 in this version of the specification.                                                                                                                                                                                                                                                         |
| SEC.WK          | 192 bits | (optional in SEC_REQ, not present in SEC_RSP) Working Key wrapped by KWK.                                                                                                                                                                                                                                                  |

|         |          |                                                                                      |
|---------|----------|--------------------------------------------------------------------------------------|
| SEC.SWK | 192 bits | (optional in SEC_REQ, not present in SEC_RSP) Subnetwork Working Key wrapped by KWK. |
|---------|----------|--------------------------------------------------------------------------------------|

Table 38 - SEC control packet types

| Name    | HDR.DO | Description                                   |
|---------|--------|-----------------------------------------------|
| SEC_REQ | 1      | Security key update request message.          |
| SEC_RSP | 0      | Security key update acknowledge to a request. |

#### 4.4.2.6.11 PCC control packet (PKT.CTYPE = 11)

This control packet is broadcast from the Base Node and relayed by all Switch Nodes to the entire Subnetwork. It is used to circulate information on a Programmed Configuration Change event, like a change of Physical layer channel/band that will be used by the Base Node at a specific time in future. The description of fields of this packet is given in Table 39, Table 40, Figure 71 and Figure 71. Further details are given in 4.6.10.

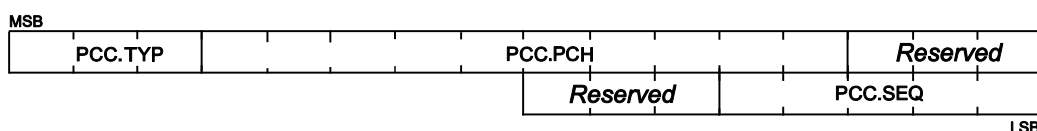


Figure 71 – PCC – PHY CH control packet structure

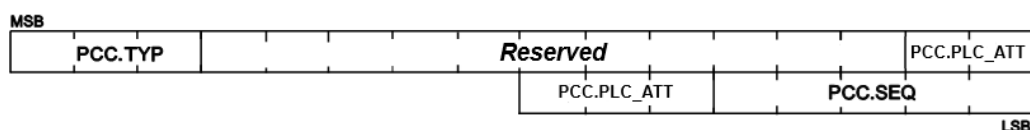


Figure 72 – PCC PLC ATT control packet structure

Table 39 - PCC control packet fields

| Name    | Length | Description                                                                                                                                                                                                                          |
|---------|--------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| PCC.TYP | 3 bits | The Programmed Configuration Change Type, encoded as:<br>0 => Change in Physical layer channel/band on a specific medium (PCC_PHY_CH)<br>1 => Change in the PLC network attenuation (PCC_PLC_ATT)<br>2-7 => Reserved for future uses |

| Name        | Length  | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |
|-------------|---------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| PCC.PCH     | 10 bits | <p>Codification of the new Physical layer channel/band that will be used by the Base Node on a specific medium.</p> <ul style="list-style-type: none"> <li>– PCC.PCH[09]=0b for the PLC channels</li> <li>– PCC.PCH[09]=1b for the SUN FSK PHY, based on clause 20 of [28,29]</li> </ul> <p>If PCC.PCH[9]=0b for the PLC channels</p> <ul style="list-style-type: none"> <li>- PCC.PCH[08]=0b reserved</li> <li>- PCC.PCH[07-00] PLC channels used. As examples; <ul style="list-style-type: none"> <li>- PCC.PCH[07-00]=01H if Channel 1 only is used.</li> <li>- PCC.PCH[07-00]=80H if Channel 8 only is used.</li> <li>- PCC.PCH[07-00]=FCH if Channel 3 to 8 are used.</li> </ul> </li> </ul> <p>If PCC.PCH[9]=1b for the RF channels</p> <ul style="list-style-type: none"> <li>- PCC.PCH[08-00] RF channels used, NumChan, according channel numbering defined on [28,29] (10.1.2.8) from 0 to TotalNumChan</li> </ul> <p>This field is only included in the PCC_PHY_CH message. Reserved and set to 0 in the remaining PCC types.</p> |
| Reserved    | n bits  | <p>Reserved for future version of this protocol. In this version this field shall be set to 0.</p> <p>If PCC.TYP = 0 (PCC_PHY_CH), n = 6 bits.</p> <p>If PCC.TYP = 1 (PCC_PLC_ATT), n = 11 bits.</p>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |
| PCC.PLC_ATT | 5 bits  | <p>PLC Network attenuation in dB (0-31).</p> <p>This field is only included in the PCC_PLC_ATT message. Reserved and set to 0 in the remaining PCC types.</p>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |
| PCC.SEQ     | 5 bits  | The Beacon Sequence number when the specified change takes effect.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           |

Table 40 - PCC control message types

| Name        | PCC.TYP | Description                                  |
|-------------|---------|----------------------------------------------|
| PCC_PHY_CH  | 000     | Physical Configuration Change                |
| PCC_PLC_ATT | 001     | PLC network attenuation Configuration Change |

#### 4.4.3 Promotion Needed PDU

If a Node is Disconnected and it does not have connectivity with any existing Switch Node, it shall send notifications to its neighbors to indicate the need for the promotion of any available Terminal



Node. Figure 73 represents the Promotion Needed MAC PDU (PNPDU) that must be sent on an irregular basis in this situation.

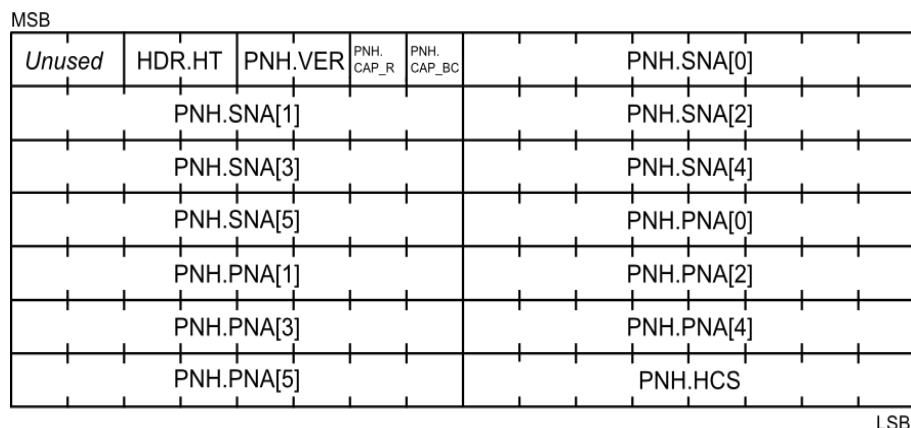


Figure 73 - Promotion Need MAC PDU

Table 41 shows the promotion need MAC PDU fields.

Table 41 - Promotion Need MAC PDU fields

| Name       | Length | Description                                                                                                                                                                                                                                                                              |
|------------|--------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Unused     | 2 bits | Unused bits which are always 0; included for alignment with MAC_H field in PPDU header (Section 3.3.3).                                                                                                                                                                                  |
| HDR.HT     | 2 bits | Header Type<br>HDR.HT = 1 for the Promotion Need MAC PDU                                                                                                                                                                                                                                 |
| PNH.VER    | 2 bits | Version of PRIME Specification: <ul style="list-style-type: none"> <li>0 – 1.3.6 PRIME</li> <li>1 – 1.4 PRIME</li> <li>2,3 – Reserved for future use</li> </ul>                                                                                                                          |
| PNH.CAP_R  | 1 bit  | Flag to define if the node supports Robust mode <ul style="list-style-type: none"> <li>0 – If the node does not support Robust mode</li> <li>1 – if the node does support Robust mode</li> </ul>                                                                                         |
| PNH.CAP_BC | 1 bit  | Flag to define if the node supports Backwards Compatibility with 1.3.6 version of the specification <ul style="list-style-type: none"> <li>0 – The node does not support Backwards Compatibility with 1.3.6</li> <li>1 – The node supports Backwards Compatibility with 1.3.6</li> </ul> |

| Name    | Length  | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |
|---------|---------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| PNH.SNA | 48 bits | <p>Subnetwork Address.</p> <p>The EUI-48 of the Base Node of the Subnetwork the Service Node is trying to connect to. FF:FF:FF:FF:FF:FF to ask for the promotion in any available Subnetwork.</p> <p>SNA[0] is the most significant byte of the OUI/IAB and SNA[5] is the least significant byte of the extension identifier, as defined in:</p> <p><a href="http://standards.ieee.org/regauth/oui/tutorials/EUI-48.html">http://standards.ieee.org/regauth/oui/tutorials/EUI-48.html</a>.</p> <p>The above notation is applicable to all EUI-48 fields in the specification.</p>                                                                 |
| PNH.PNA | 48 bits | Promotion Need Address. The EUI-48 of the Node that needs the promotion. It is the EUI-48 of the transmitter.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |
| PNH.HCS | 8 bits  | <p>Header Check Sequence. A field for detecting errors in the header. The transmitter shall calculate the PNH.HCS of the first 13 bytes of the header and insert the result into the PNH.HCS field (the last byte of the header). It shall be calculated as the remainder of the division (Modulo 2) of the polynomial <math>M(x) \cdot x^8</math> by the generator polynomial <math>g(x) = x^8 + x^2 + x + 1</math>. <math>M(x)</math> is the input polynomial, which is formed by the bit sequence of the header excluding the PNH.HCS field, and the msb of the bit sequence is the coefficient of the highest order of <math>M(x)</math>.</p> |

2459

2460 As it is always transmitted by unsynchronized Nodes and, therefore, prone to creating collisions, it is a special  
 2461 reduced size header.

#### 2462 4.4.4 Beacon PDU

2463 Beacon PDU (BPDU) is transmitted by every Switch device on the Subnetwork, including the Base Node. The  
 2464 purpose of this PDU is to circulate information on MAC frame structure and therefore channel access to all  
 2465 devices that are part of this Subnetwork. The BPDU is transmitted at definite fixed intervals of time and is  
 2466 also used as a synchronization mechanism by Service Nodes. Figure 74 below shows contents of a beacon  
 2467 transmitted by the Base Node and each Switch Device. It is important to remark that the BCN.HOP\_POS field  
 2468 may be only transmitted on the RF medium, so the beacon format may be different on the two media.

2469

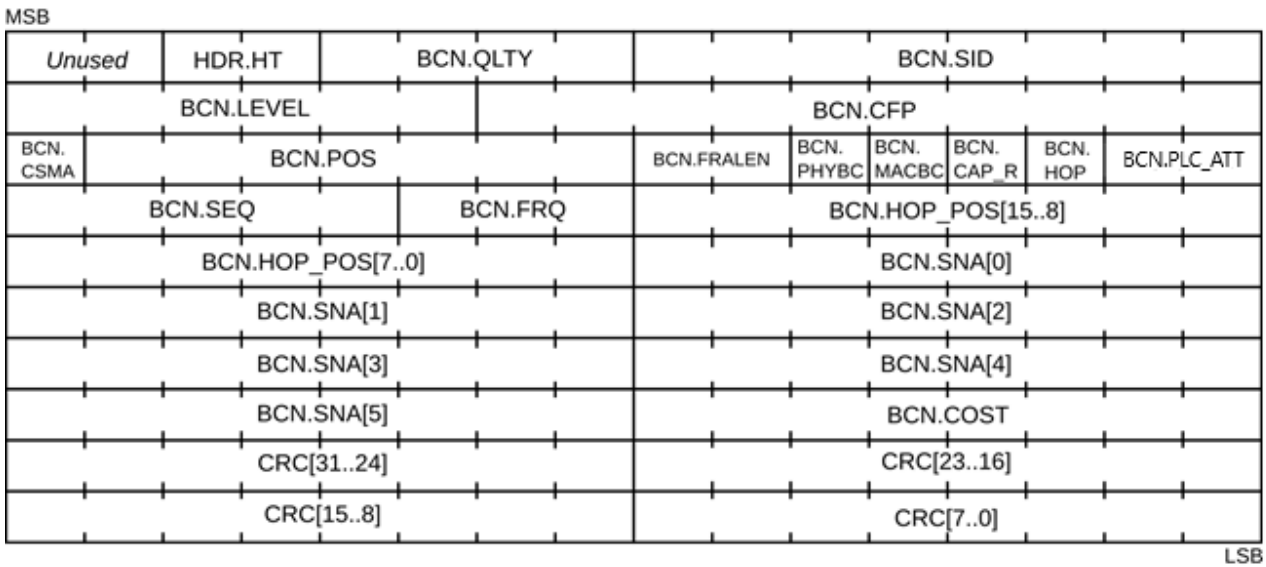


Figure 74 – Beacon PDU structure

Table 42 shows the beacon PDU fields.

Table 42 - Beacon PDU fields

| Name   | Length | Description                                                                                                    |
|--------|--------|----------------------------------------------------------------------------------------------------------------|
| Unused | 2 bits | Unused bits which are always 0; included for alignment with MAC_H field in PPDU header (Fig 7, Section 3.3.3). |
| HDR.HT | 2 bits | Header Type<br><br>HDR.HT = 2 for Beacon PDU                                                                   |

| Name        | Length  | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |
|-------------|---------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| BCN.QLTY    | 4 bits  | <p>Quality of round-trip connectivity from this Switch Node to the Base Node. ,with the following meaning:</p> <p>BCN.QLTY = 0 if <math>1/2 &lt; \text{rtdp} \leq 1</math><br/> BCN.QLTY = 1 if <math>3/8 &lt; \text{rtdp} \leq 1/2</math><br/> BCN.QLTY = 2 if <math>1/4 &lt; \text{rtdp} \leq 3/8</math><br/> BCN.QLTY = 3 if <math>3/16 &lt; \text{rtdp} \leq 1/4</math><br/> BCN.QLTY = 4 if <math>1/8 &lt; \text{rtdp} \leq 3/16</math><br/> BCN.QLTY = 5 if <math>3/32 &lt; \text{rtdp} \leq 1/8</math><br/> BCN.QLTY = 6 if <math>1/16 &lt; \text{rtdp} \leq 3/32</math><br/> BCN.QLTY = 7 if <math>3/64 &lt; \text{rtdp} \leq 1/16</math><br/> BCN.QLTY = 8 if <math>1/32 &lt; \text{rtdp} \leq 3/64</math><br/> BCN.QLTY = 9 if <math>3/128 &lt; \text{rtdp} \leq 1/32</math><br/> BCN.QLTY = 10 if <math>1/64 &lt; \text{rtdp} \leq 3/128</math><br/> BCN.QLTY = 11 if <math>3/256 &lt; \text{rtdp} \leq 1/64</math><br/> BCN.QLTY = 12 if <math>1/128 &lt; \text{rtdp} \leq 3/256</math><br/> BCN.QLTY = 13 if <math>3/512 &lt; \text{rtdp} \leq 1/128</math><br/> BCN.QLTY = 14 if <math>1/256 &lt; \text{rtdp} \leq 3/512</math><br/> BCN.QLTY = 15 if <math>\text{rtdp} \leq 1/256</math></p> <p>where:</p> <p>rtdp = Round Trip Drop Probability. Probability for a packet to be dropped when it is supposed to go downlink and be answered uplink (or the other way around) between the Base Node and the Switch Node.</p> <p>It is up to the manufacturer how to detect it, and It doesn't have to be very accurate, just an estimation. As a guideline, ALV packets can be used to calculate this field.</p> |
| BCN.SID     | 8 bits  | Switch identifier of transmitting Switch                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |
| BCN.LEVEL   | 6 bits  | Hierarchy of transmitting Switch in Subnetwork                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 |
| BCN.CFP     | 10 bits | CFP length in symbols.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |
| BCN.CSMA    | 1 bit   | <p>PLC CSMA/CA Algorithm used in the Subnetwork.</p> <ul style="list-style-type: none"> <li>0 – CSMA/CA Algorithm 1 (i.e. v1.4)</li> <li>1 – CSMA/CA Algorithm 2 (i.e. v1.3.6, as in backward compatibility mode)</li> </ul>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |
| BCN.POS     | 7 bits  | Position of this beacon in symbols from the beginning of the frame.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            |
| BCN.FRA_LEN | 2 bits  | <p>Length of the frame.</p> <ul style="list-style-type: none"> <li>0 - 276 symbols</li> <li>1 - 552 symbols</li> <li>2 - 828 symbols</li> <li>3 - 1104 symbols</li> </ul>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |

| Name        | Length | Description                                                                                                                                                                                                                                                                                          |
|-------------|--------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| BCN.PHYBC   | 1 bit  | PHY backwards compatibility mode:<br>0 – The network is working in normal mode.<br>1 - The network is working on PHY backwards compatibility mode, all the nodes that need to send Type B PHY Frames shall use PHY backwards compatible frames.                                                      |
| BCN.MACBC   | 1 bit  | MAC backward compatibility mode:<br>0 – The network is working in 1.4 mode.<br>1 – The network is working in MAC backward compatibility mode, see section 4.9 for details.                                                                                                                           |
| BCN.CAP_R   | 1 bit  | Robust Mode Capable<br>1 – The device is able to transmit/receive robust mode frames.<br>0 – The device is not able to transmit/receive robust mode frames.                                                                                                                                          |
| BCN.HOP     | 1 bit  | Indicates the presence of the BCN.HOP_POS field needed for channel hopping synchronization:<br>0 – The BCN.HOP_POS field is not present.<br>1 – The BCN.HOP_POS field is present.                                                                                                                    |
| BCN.PLC_ATT | 2 bits | Indicates the PLC attenuation range used in the network:<br>00b – 0-7 dB attenuation range.<br>01b – 8-15 dB attenuation range.<br>10b – 16-23 dB attenuation range.<br>11b – 24-31 dB attenuation range.                                                                                            |
| BCN.SEQ     | 5 bits | Sequence number of this BPDU in super frame. Incremented for every beacon the Base Node sends and is propagated by Switch through its BPDU such that entire Subnetwork has the same notion of sequence number at a given time.                                                                       |
| BCN.FRQ     | 3 bits | Transmission frequency of this BPDU. Values are interpreted as follows:<br><br>0 = 1 beacon every frame<br>1 = 1 beacon every 2 frames<br>2 = 1 beacon every 4 frames<br>3 = 1 beacon every 8 frames<br>4 = 1 beacon every 16 frames<br>5 = 1 beacon every 32 frames<br>6 = Reserved<br>7 = Reserved |

| Name        | Length  | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |
|-------------|---------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| BCN.HOP_POS | 16 bits | This field indicates the current macHoppingSequencePosition required for channel hopping synchronization. This field is optional and only present when indicated by the BCN.HOP field.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |
| BCN.SNA     | 48 bits | Subnetwork identifier in which the Switch transmitting this BPDU is located                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |
| BCN.COST    | 8 bits  | <p>Total cost from the transmitting Switch Node to the Base Node. The cost of a single hop is calculated based on modulation scheme used on that hop in both downlink and uplink direction. For PLC, values are derived as follows:</p> <p>8PSK = 0<br/> QPSK = 0<br/> BPSK = 0<br/> 8PSK_F = 0<br/> QPSK_F = 1<br/> BPSK_F = 2<br/> QPSK_R = 4<br/> BPSK_R = 8</p> <p>For RF, the link cost value to be used is reported in macRFLinkCost (see Table 106). The Base Node shall transmit in its beacon a BCN.COST of 0. A Switch Node shall transmit in its beacon the value of BCN.COST received from its upstream Switch Node, plus the cost of the upstream hop to its upstream Switch, calculated as the addition of both uplink and downlink costs. When this value is larger than what can be held, the maximum value of BCN.COST shall be used.</p> |
| CRC         | 32 bits | The CRC shall be calculated with the same algorithm as the one defined for the CRC field of the MAC PDU (see section 4.4.2.4 for details). Since the SNA is present in the payload there is no concatenation of SNA and GPDU in this case. For CRC calculation the field CRC is set to the constant 0x00010400. The CRC shall be calculated over the whole BPDU, including constant CRC field                                                                                                                                                                                                                                                                                                                                                                                                                                                              |

2474

2475 The BPDU is also used to detect when the uplink Switch is no longer available either by a change in the  
2476 characteristics of the medium or because of failure etc. If a Service Node fails to receive all the expected  
2477 beacons during Nmiss-beacon superframes it shall declare the link to its Switch as unusable. The Service Node  
2478 shall stop sending beacons itself if it is acting as a Switch. It shall close all existing MAC connections. The  
2479 Service Node then enters the initial Disconnected state and searches for a Subnetwork join. This mechanism  
2480 complements the Keep-Alive mechanism which is used by a Base Node and its switches to determine when  
2481 a Service Node is lost.

## 4.5 MAC Service Access Point

### 4.5.1 General

The MAC service access point provides several primitives to allow the Convergence layer to interact with the MAC layer. This section aims to explain how the MAC may be used. An implementation of the MAC may not use all the primitives listed here; it may use other primitives; or it may have a function-call based interface rather than message-passing, etc. These are all implementation issues which are beyond the scope of this specification.

The .request primitives are passed from the CL to the MAC to request the initiation of a service. The .indication and .confirm primitives are passed from the MAC to the CL to indicate an internal MAC event that is significant to the CL. This event may be logically related to a remote service request or may be caused by an event internal to the local MAC. The .response primitive is passed from the CL to the MAC to provide a response to a .indication primitive. Thus, the four primitives are used in pairs, the pair .request and .confirm and the pair .indication and .response. This is shown in Figure 75, Figure 76, Figure 77 and Figure 78.

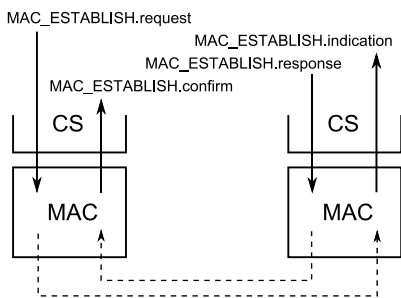


Figure 75 – Establishment of a Connection

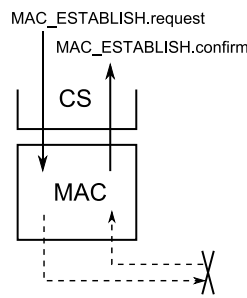


Figure 76 – Failed establishment of a Connection

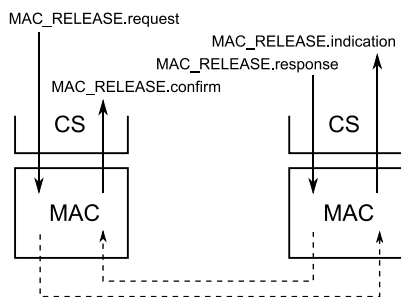


Figure 77 – Release of a Connection

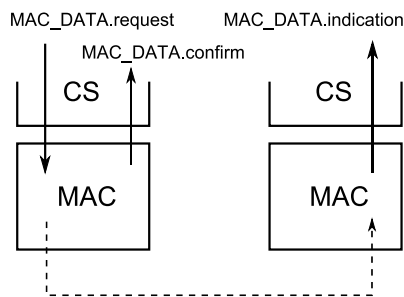


Figure 78- Transfer of Data

Table 43 represents the list of available primitives in the MAC-SAP:

Table 43 – List of MAC primitives

| Service Node primitives  | Base Node primitives     |
|--------------------------|--------------------------|
| MAC_ESTABLISH.request    | MAC_ESTABLISH.request    |
| MAC_ESTABLISH.indication | MAC_ESTABLISH.indication |
| MAC_ESTABLISH.response   | MAC_ESTABLISH.response   |

| Service Node primitives | Base Node primitives   |
|-------------------------|------------------------|
| MAC_ESTABLISH.confirm   | MAC_ESTABLISH.confirm  |
| MAC_RELEASE.request     | MAC_RELEASE.request    |
| MAC_RELEASE.indication  | MAC_RELEASE.indication |
| MAC_RELEASE.response    | MAC_RELEASE.response   |
| MAC_RELEASE.confirm     | MAC_RELEASE.confirm    |
| MAC_JOIN.request        | MAC_JOIN.request       |
| MAC_JOIN.Response       | MAC_JOIN.response      |
| MAC_JOIN.indication     | MAC_JOIN.indication    |
| MAC_JOIN.confirm        | MAC_JOIN.confirm       |
| MAC_LEAVE.request       | MAC_LEAVE.request      |
| MAC_LEAVE.indication    | MAC_LEAVE.indication   |
| MAC_LEAVE.confirm       | MAC_LEAVE.confirm      |
| MAC_DATA.request        | MAC_REDIRECT.response  |
| MAC_DATA.confirm        | MAC_DATA.request       |
| MAC_DATA.indication     | MAC_DATA.confirm       |
|                         | MAC_DATA.indication    |

## 2497 4.5.2 Service Node and Base Node signalling primitives

### 2498 4.5.2.1 General

2499 The following subsections describe primitives which are available in both the Service Node and Base Node  
2500 MAC-SAP. These are signaling primitives only and used for establishing and releasing MAC connections.

### 2501 4.5.2.2 MAC\_ESTABLISH

#### 2502 4.5.2.2.1 General

2503 The MAC\_ESTABLISH primitives are used to manage a connection establishment.

#### 2504 4.5.2.2.2 MAC\_ESTABLISH.request

2505 The MAC\_ESTABLISH.request primitive is passed to the MAC layer entity to request the connection  
2506 establishment.

2507 The semantics of this primitive are as follows:

2508 *MAC\_ESTABLISH.request{EUI-48, Type, Data, DataLength, ARQ, CfBytes, AE}*



2509 The *EUI-48* parameter of this primitive is used to specify the address of the Node to which this connection  
 2510 will be addressed. The MAC will internally transfer this to an address used by the MAC layer. When the CL of  
 2511 a Service Node wishes to connect to the Base Node, it uses the EUI-48 00:00:00:00:00:00. However, when  
 2512 the CL of a Service Node wishes to connect to another Service Node on the Subnetwork, it uses the EUI-48 of  
 2513 that Service Node. This will then trigger a direct connection establishment. However, whether a normal or a  
 2514 directed connection is established is transparent to the Service Node MAC SAP. As the EUI-48 of the Base  
 2515 Node is the SNA, the connection could also be requested from the Base Node using the SNA.

2516 The *Type* parameter is an identifier used to define the type of the Convergence layer that should be used for  
 2517 this connection (see Annex E). This parameter is 1 byte long and will be transmitted in the CON.TYPE field of  
 2518 the connection request.

2519 The *Data* parameter is a general purpose buffer to be interchanged for the negotiation between the local CL  
 2520 and the remote CL. This parameter will be transmitted in the CON.DATA field of the connection request.

2521 The *DataLength* parameter is the length of the *Data* parameter in bytes.

2522 The *ARQ* parameter indicates whether or not the ARQ mechanism should be used for this connection. It is a  
 2523 Boolean type with a value of true indicating that ARQ will be used.

2524 The *CfBytes* parameter is used to indicate whether or not the connection should use the contention or  
 2525 contention-free channel access scheme. When *CfBytes* is zero, contention-based access should be used.  
 2526 When *CfBytes* is not zero, it indicates how many bytes per frame should be allocated to the connection using  
 2527 CFP packets.

2528 The *AE* parameter indicates whether or not the information transmitted in this connection is encrypted. It is  
 2529 a Boolean type with a value of true indicating that encryption will be used.

#### 2530 **4.5.2.2.3 MAC\_ESTABLISH.indication**

2531 The MAC\_ESTABLISH.indication is passed from the MAC layer to indicate that a connection establishment  
 2532 was initiated by a remote Node.

2533 The semantics of this primitive are as follows:

2534 *MAC\_ESTABLISH.indication*{*ConHandle*, *EUI-48*, *Type*, *Data*, *DataLength*, *CfBytes*, *AE*}

2535 The *ConHandle* is a unique identifier interchanged to uniquely identify the connection being indicated. It has  
 2536 a valid meaning only in the MAC SAP, used to have a reference to this connection between different  
 2537 primitives.

2538 The *EUI-48* parameter indicates which device on the Subnetwork wishes to establish a connection.

2539 The *Type* parameter is an identifier used to define the type of the Convergence layer that should be used for  
 2540 this connection. This parameter is 1 byte long and it is received in the CON.TYPE field of the connection  
 2541 request.

2542 The *Data* parameter is a general purpose buffer to be interchanged for the negotiation between the remote  
 2543 CL and the local CL. This parameter is received in the CON.DATA field of the connection request.

- 2544 The *DataLength* parameter is the length of the Data parameter in bytes.
- 2545 The *CfBytes* parameter is used to indicate if the connection should use the contention or contention-free  
2546 channel access scheme. When *CfBytes* is zero, contention-based access will be used. When *CfBytes* is not  
2547 zero, it indicates how many bytes per frame the connection would like to be allocated.
- 2548 The *AE* parameter indicates whether or not the information transmitted in this connection is encrypted. It is  
2549 a Boolean type with a value of true indicating that encryption will be used.
- 2550 **4.5.2.2.4 MAC\_ESTABLISH.response**
- 2551 The MAC\_ESTABLISH.response is passed to the MAC layer to respond with a MAC\_ESTABLISH.indication.
- 2552 The semantics of this primitive are as follows:
- 2553 *MAC\_ESTABLISH.response*{*ConHandle*, *Answer*, *Data*, *DataLength*, *AE*}
- 2554 The *ConHandle* parameter is the same as the one that was received in the MAC\_ESTABLISH.indication.
- 2555 The *Answer* parameter is used to notify the MAC of the action to be taken for this connection establishment.  
2556 This parameter may have one of the values in Table 44.
- 2557 The *Data* parameter is a general purpose buffer to be interchanged for the negotiation between the remote  
2558 CL and the local CL. This parameter is received in the CON.DATA field of the connection response.
- 2559 The *DataLength* parameter is the length of the Data parameter in bytes.
- 2560 Data may be passed to the caller even when the connection is rejected, i.e. Answer has the value 1. The data  
2561 may then optionally contain more information as to why the connection was rejected.
- 2562 The *AE* parameter indicates whether or not the information transmitted in this connection is encrypted. It is  
2563 a Boolean type with a value of true indicating that encryption will be used.

2564 **Table 44 – Values of the *Answer* parameter in MAC\_ESTABLISH.response primitive**

| <i>Answer</i>     | Description                               |
|-------------------|-------------------------------------------|
| <i>Accept</i> = 0 | The connection establishment is accepted. |
| <i>Reject</i> = 1 | The connection establishment is rejected. |

- 2565 **4.5.2.2.5 MAC\_ESTABLISH.confirm**
- 2566 The MAC\_ESTABLISH.confirm is passed from the MAC layer as the remote answer to a  
2567 MAC\_ESTABLISH.request.
- 2568 The semantics of this primitive are as follows:
- 2569 *MAC\_ESTABLISH.confirm*{*ConHandle*, *Result*, *EUI-48*, *Type*, *Data*, *DataLength*, *AE*}

2570 The *ConHandle* is a unique identifier to uniquely identify the connection being indicated. It has a valid  
 2571 meaning only in the MAC SAP, used to have a reference to this connection between different primitives. The  
 2572 value is only valid if the *Result* parameter is 0.

2573 The *Result* parameter indicates the result of the connection establishment process. It may have one of the  
 2574 values in Table 45 .

2575 The *EUI-48* parameter indicates which device on the Subnetwork accepted or refused to establish a  
 2576 connection.

2577 The *Type* parameter is an identifier used to define the type of the Convergence layer that should be used for  
 2578 this connection. This parameter is 1 byte long and it is received in the CON.TYPE field of the connection  
 2579 request

2580 The *Data* parameter is a general purpose buffer to be interchanged for the negotiation between the remote  
 2581 CL and the local CL. This parameter is received in the CON.DATA field of the connection response.

2582 The *DataLength* parameter is the length of the Data parameter in bytes.

2583 Data may be passed to the caller even when the connection is rejected, i.e. Result has the value 1. The data  
 2584 may then optionally contain more information as to why the connection was rejected.

2585 The *AE* parameter indicates whether or not the information transmitted in this connection is encrypted. It is  
 2586 a Boolean type with a value of true indicating that encryption will be used.

2587 **Table 45 – Values of the *Result* parameter in MAC\_ESTABLISH.confirm primitive**

| <b><i>Result</i></b>       | <b>Description</b>                                                                   |
|----------------------------|--------------------------------------------------------------------------------------|
| <i>Success</i> = 0         | The connection establishment was successful.                                         |
| <i>Reject</i> = 1          | The connection establishment failed because it was rejected by the remote Node.      |
| <i>Timeout</i> = 2         | The connection establishment process timed out.                                      |
| <i>No bandwidth</i> = 3    | There is insufficient available bandwidth to accept this contention-free connection. |
| <i>No Such Device</i> = 4  | A device with the destination address cannot be found.                               |
| <i>Redirect failed</i> =5  | The Base Node attempted to perform a redirect which failed.                          |
| <i>Not Registered</i> = 6  | The Service Node is not registered.                                                  |
| <i>No More LCIDs</i> = 7   | All available LCIDs have been allocated.                                             |
| <i>Unsupported SP</i> = 14 | Device doesn't support SP > 0 but asked for an encrypted connection establishment    |

### 2588 **4.5.2.3 MAC\_RELEASE**

#### 2589 **4.5.2.3.1 General**

2590 The MAC\_RELEASE primitives are used to release a connection.

#### 2591 4.5.2.3.2 MAC\_RELEASE.request

2592 The MAC\_RELEASE.request is a primitive used to initiate the release process of a connection.

2593 The semantics of this primitive are as follows:

2594 *MAC\_RELEASE.request{ConHandle}*

2595 The *ConHandle* parameter specifies the connection to be released. This handle is the one that was obtained  
2596 during the MAC\_ESTABLISH primitives.

#### 2597 4.5.2.3.3 MAC\_RELEASE.indication

2598 The MAC\_RELEASE.indication is a primitive used to indicate that a connection is being released. It may be  
2599 released because of a remote operation or because of a connectivity problem.

2600 The semantics of this primitive are as follows:

2601 *MAC\_RELEASE.indication{ConHandle, Reason}*

2602 The *ConHandle* parameter specifies the connection being released. This handle is the one that was obtained  
2603 during the MAC\_ESTABLISH primitives.

2604 The *Reason* parameter may have one of the values given in Table 46.

2605 Table 46 – Values of the *Reason* parameter in MAC\_RELEASE.indication primitive

| <i>Reason</i>      | Description                                                    |
|--------------------|----------------------------------------------------------------|
| <i>Success</i> = 0 | The connection release was initiated by a remote service.      |
| <i>Error</i> = 1   | The connection was released because of a connectivity problem. |

#### 2606 4.5.2.3.4 MAC\_RELEASE.response

2607 The MAC\_RELEASE.response is a primitive used to respond to a connection release process.

2608 The semantics of this primitive are as follows:

2609 *MAC\_RELEASE.response{ConHandle, Answer}*

2610 The *ConHandle* parameter specifies the connection being released. This handle is the one that was obtained  
2611 during the MAC\_ESTABLISH primitives.

2612 The *Answer* parameter may have one of the values given in Table 47 This parameter may not have the value  
2613 “*Reject* = 1” because a connection release process cannot be rejected.

2614 Table 47 – Values of the *Answer* parameter in MAC\_RELEASE.response primitive

| <i>Answer</i>     | Description                         |
|-------------------|-------------------------------------|
| <i>Accept</i> = 0 | The connection release is accepted. |

2615

2616 After sending the MAC\_RELEASE.response the ConHandle is no longer valid and should not be used.

#### 2617 4.5.2.3.5 MAC\_RELEASE.confirm

2618 The MAC\_RELEASE.confirm primitive is used to confirm that the connection release process has finished.

2619 The semantics of this primitive are as follows:

2620 *MAC\_RELEASE.confirm{ConHandle, Result}*

2621 The *ConHandle* parameter specifies the connection released. This handle is the one that was obtained during  
2622 the MAC\_ESTABLISH primitives.

2623 The *Result* parameter may have one of the values given in Table 48

2624 Table 48 – Values of the Result parameter in MAC\_RELEASE.confirm primitive

| <b>Result</b>             | <b>Description</b>                        |
|---------------------------|-------------------------------------------|
| <i>Success = 0</i>        | The connection release was successful.    |
| <i>Timeout = 2</i>        | The connection release process timed out. |
| <i>Not Registered = 6</i> | The Service Node is no longer registered. |

2625

2626 After the reception of the MAC\_RELEASE.confirm the ConHandle is no longer valid and should not be used.

#### 2627 4.5.2.4 MAC\_JOIN

##### 2628 4.5.2.4.1 General

2629 The MAC\_JOIN primitives are used to join to a broadcast or multicast connection and allow the reception of  
2630 such packets.

##### 2631 4.5.2.4.2 MAC\_JOIN.request

2632 The MAC\_JOIN.request primitive is used:

- 2633 • By all Nodes : to join broadcast traffic of a specific CL and start receiving these packets
- 2634 • By Service Nodes : to join a particular multicast group
- 2635 • By Base Node : to invite a Service Node to join a particular multicast group

2636 Depending on which device makes the join-request, this SAP can have two different variants. First variant  
2637 shall be used on Base Nodes and second on Service Nodes:

2638 The semantics of this primitive are as follows:

2639 *MAC\_JOIN.request{Broadcast, ConHandle, EUI-48, Type, Data, DataLength, AE}*

2640 *MAC\_JOIN.request{Broadcast, Type, Data, DataLength, AE }*

2641 The *Broadcast* parameter specifies whether the JOIN operation is being performed for a broadcast  
2642 connection or for a multicast operation. It should be 1 for a broadcast operation and 0 for a multicast  
2643 operation. In case of broadcast operation, EUI-48, Data, DataLength are not used.

2644 ConHandle indicates the handle to be used with for this multicast join. In case of first join request for a new  
2645 multicast group, ConHandle will be set to 0. For any subsequent EUI additions to an existing multicast group,  
2646 ConHandle will serve as index to respective multicast group.

2647 The EUI-48 parameter is used by the Base Node to specify the address of the Node to which this join request  
2648 will be addressed. The MAC will internally transfer this to an address used by the MAC layer. When the CL of  
2649 a Service Node initiates the request, it uses the EUI-48 00:00:00:00:00:00.

2650 The Type parameter defines the type of the Convergence layer that will send/receive the data packets. This  
2651 parameter is 1 byte long and will be transmitted in the MUL.TYPE field of the join request.

2652 The Data parameter is a general purpose buffer to be interchanged for the negotiation between the remote  
2653 CL and the local CL. This parameter is received in the MUL.DATA field of the connection request. In case the  
2654 CL supports several multicast groups, this Data parameter will be used to uniquely identify the group

2655 The DataLength parameter is the length of the Data parameter in bytes.

2656 If Broadcast is 1, the MAC will immediately issue a MAC\_JOIN.confirm primitive since it does not need to  
2657 perform any end-to-end operation. For a multicast operation the MAC\_JOIN.confirm is only sent once  
2658 signaling with the uplink Service Node/Base Node is complete.

2659 The AE parameter indicates whether or not the information transmitted in this connection is encrypted. It is  
2660 a Boolean type with a value of true indicating that encryption will be used.

#### 2661 4.5.2.4.3 MAC\_JOIN.confirm

2662 The MAC\_JOIN.confirm primitive is received to confirm that the MAC\_JOIN.request operation has finished.

2663 The semantics of this primitive are as follows:

2664 
$$\text{MAC\_JOIN.confirm}\{\text{ConHandle}, \text{Result}, \text{AE}\}$$

2665 The *ConHandle* is a unique identifier to uniquely identify the connection being indicated. It has a valid  
2666 meaning only in the MAC SAP, used to have a reference to this connection between different primitives. The  
2667 value is only valid if the *Result* parameter is 0. When the MAC receives packets on this connection, they will  
2668 be passed upwards using the MAC\_DATA.indication primitive with this *ConHandle*.

2669 The Result parameter indicates the result of multicast group join process. It may have one of the values given  
2670 in Table 49.

2671 The AE parameter indicates whether or not the information transmitted in this connection is encrypted. It is  
2672 a Boolean type with a value of true indicating that encryption will be used.

2673 **Table 49 – Values of the *Result* parameter in MAC\_JOIN.confirm primitive**

| <b>Result</b>                  | <b>Description</b>                                                                                  |
|--------------------------------|-----------------------------------------------------------------------------------------------------|
| <i>Success = 0</i>             | The connection establishment was successful.                                                        |
| <i>Reject = 1</i>              | The connection establishment failed because it was rejected by the upstream Service Node/Base Node. |
| <i>Timeout = 2</i>             | The connection establishment process timed out.                                                     |
| <i>Unsupported<br/>SP = 14</i> | Device doesn't support SP > 0 but asked to join a multicast group with an encrypted connection      |

#### 2674 4.5.2.4.4 MAC\_JOIN.indication

2675 On the Base Node, the MAC\_JOIN.indication is passed from the MAC layer to indicate that a multicast group  
 2676 join was initiated by a Service Node. On a Service Node, it is used to indicate that the Base Node is inviting to  
 2677 join a multicast group.

2678 Depending on device type, this primitive shall have two variants. The first variant below shall be used in Base  
 2679 Nodes and the second variant is for Service Nodes:

2680 *MAC\_JOIN.indication{ConHandle, EUI-48, Type, Data, DataLength, AE}*

2681 *MAC\_JOIN.indication{ConHandle, Type, Data, DataLength, AE}*

2682 The *ConHandle* is a unique identifier interchanged to uniquely identify the multicast group being indicated.  
 2683 It has a valid meaning only in the MAC SAP, used to have a reference to this connection between different  
 2684 primitives.

2685 The *EUI-48* parameter indicates which device on the Subnetwork wishes to establish a connection.

2686 The *Type* parameter is an identifier used to define the type of the Convergence layer that should be used for  
 2687 this request. This parameter is 1 byte long and it is received in the MUL.TYPE field of the connection request.

2688 The *Data* parameter is a general purpose buffer to be interchanged for the negotiation between the remote  
 2689 CL and the local CL. This parameter is received in the MUL.DATA field of the connection request.

2690 The *DataLength* parameter is the length of the Data parameter in bytes.

2691 The *AE* parameter indicates whether or not the information transmitted in this connection is encrypted. It is  
 2692 a Boolean type with a value of true indicating that encryption will be used.

#### 2693 4.5.2.4.5 MAC\_JOIN.response

2694 The MAC\_JOIN.response is passed to the MAC layer to respond with a MAC\_JOIN.indication. Depending on  
 2695 device type, this primitive could have either of the two forms given below. The first one shall be used in  
 2696 Service Node and the second one in Base Node implementations.

2697 The semantics of this primitive are as follows:

2698 *MAC\_JOIN.response{ConHandle, Answer, AE}*

2699 *MAC\_JOIN.response (ConHandle, EUI, Answer, AE)*

2700 The *ConHandle* parameter is the same as the one that was received in the MAC\_JOIN.indication.

2701 *EUI* is the EUI-48 of Service Node that requested the multicast group join.

2702 The *Answer* parameter is used to notify the MAC of the action to be taken for this join request. This parameter  
2703 may have one of the values depicted below.

2704 The *AE* parameter indicates whether or not the information transmitted in this connection is encrypted. It is  
2705 a Boolean type with a value of true indicating that encryption will be used.

2706 **Table 50 – Values of the *Answer* parameter in MAC\_ESTABLISH.response primitive**

| Answer     | Description                           |
|------------|---------------------------------------|
| Accept = 0 | The multicast group join is accepted. |
| Reject = 1 | The multicast group join is rejected. |

#### 2707 **4.5.2.5 MAC\_LEAVE**

##### 2708 **4.5.2.5.1 General**

2709 The MAC\_LEAVE primitives are used to leave a broadcast or multicast connection.

##### 2710 **4.5.2.5.2 MAC\_LEAVE.request**

2711 The MAC\_LEAVE.request primitive is used to leave a multicast or broadcast traffic. Depending on device type,  
2712 this primitive could have either of the two forms given below. The first one shall be used in Service Node and  
2713 the second one in Base Node implementations.

2714 The semantics of this primitive are as follows:

2715 *MAC\_LEAVE.request{ConHandle}*

2716 *MAC\_LEAVE.request{ConHandle, EUI}*

2717 The *ConHandle* parameter specifies the connection to be left. This handle is the one that was obtained during  
2718 the MAC\_JOIN primitives.

2719 *EUI* is the EUI-48 of Service Node to remove from multicast group.

##### 2720 **4.5.2.5.3 MAC\_LEAVE.confirm**

2721 The MAC\_LEAVE.confirm primitive is received to confirm that the MAC\_LEAVE.request operation has  
2722 finished.

2723 The semantics of this primitive are as follows:

2724 *MAC\_LEAVE.confirm{ConHandle, Result}*



2725 The *ConHandle* parameter specifies the connection released. This handle is the one that was obtained during  
2726 the MAC\_JOIN primitives.

2727 The *Result* parameter may have one of the values in Table 51.

2728 **Table 51 – Values of the *Result* parameter in MAC\_LEAVE.confirm primitive**

| <b><i>Result</i></b> | <b>Description</b>                      |
|----------------------|-----------------------------------------|
| <i>Success</i> = 0   | The connection leave was successful.    |
| <i>Timeout</i> = 2   | The connection leave process timed out. |

2729

2730 After the reception of the MAC\_LEAVE.confirm, the ConHandle is no longer valid and should not be used.

#### 2731 **4.5.2.5.4 MAC\_LEAVE.indication**

2732 The MAC\_LEAVE.indication primitive is used to leave a multicast or broadcast traffic. Depending on device  
2733 type, this primitive could have either of the two forms given below. The first one shall be used in Service  
2734 Node and the second one in Base Node implementations.

2735 The semantics of this primitive are as follows:

2736 *MAC\_LEAVE.indication*{*ConHandle*}

2737 *MAC\_LEAVE.indication*{*ConHandle*, *EUI*}

2738 The ConHandle parameter is the same as that received in MAC\_JOIN.confirm or MAC\_JOIN.indication. This  
2739 handle is the one that was obtained during the MAC\_JOIN primitives.

2740 *EUI* is the EUI-48 of Service Node to remove from multicast group.

### 2741 **4.5.3 Base Node signalling primitives**

#### 2742 **4.5.3.1 General**

2743 This section specifies MAC-SAP primitives that are only available in the Base Node.

#### 2744 **4.5.3.2 MAC\_REDIRECT.response**

2745 The MAC\_REDIRECT.response primitive is used to answer to a MAC\_ESTABLISH.indication and redirects the  
2746 connection from the Base Node to another Service Node on the Subnetwork.

2747 The semantics of this primitive are as follows:

2748 *MAC\_REDIRECT.reponse*{*ConHandle*, *EUI-48*, *Data*, *DataLength*}

2749 The *ConHandle* is the one passed in the MAC\_ESTABLISH.indication primitive to which it is replying.

2750 *EUI-48* indicates the Service Node to which this connection establishment should be forwarded. The Base  
2751 Node should perform a direct connection setup between the source of the connection establishment and the  
2752 Service Node indicated by *EUI-48*.

2753 The *Data* parameter is a general purpose buffer to be interchanged for the negotiation between the remote  
2754 CL and the Base Node CL. This parameter is received in the CON.DATA field of the connection request.

2755 The *DataLength* parameter is the length of the Data parameter in bytes.

2756 Once this primitive has been used, the ConHandle is no longer valid.

## 2757 4.5.4 Service and Base Nodes data primitives

### 2758 4.5.4.1 General

2759 The following subsections describe how a Service Node or Base Node passes data between the Convergence  
2760 layer and the MAC layer.

### 2761 4.5.4.2 MAC\_DATA.request

2762 The MAC\_DATA.request primitive is used to initiate the transmission process of data over a connection.

2763 The semantics of the primitive are as follows:

2764 *MAC\_DATA.request{ConHandle, Data, DataLength, Priority, TimeReference}*

2765 The *ConHandle* parameter specifies the connection to be used for the data transmission. This handle is the  
2766 one that was obtained during the connection establishment primitives.

2767 The *Data* parameter is a buffer of octets that contains the CL data to be transmitted through this connection.

2768 The *DataLength* parameter is the length of the *Data* parameter in octets.

2769 *Priority* indicates the priority of the data to be sent when using the PLC CSMA access scheme, i.e. the  
2770 parameter only has meaning when the connection was established with CfBytes = 0. When SUN FSK profile  
2771 is supported and a data packet is transmitted over RF, this value is ignored by the RF CSMA.

2772 The *TimeReference* parameter is the time reference to interchange with the data. This *TimeReference*  
2773 parameter is optional; it is possible not sending any time reference. From the primitive point of view the act  
2774 of not including a time reference will be considered a *NULL* time reference. The way to interchange this  
2775 parameter in the primitive not loosing precision and its absolute meaning are specific to the implementation.

### 2776 4.5.4.3 MAC\_DATA.confirm

2777 The MAC\_DATA.confirm primitive is used to confirm that the transmission process of the data has completed.

2778 The semantics of the primitive are as follows:

2779 *MAC\_DATA.confirm{ConHandle, Data, Result}*

2780 The *ConHandle* parameter specifies the connection that was used for the data transmission. This handle is  
2781 the one that was obtained during the connection establishment primitives.

2782 The *Data* parameter is a buffer of octets that contains the CL data that where to be transmitted through this  
2783 connection.

2784 The Result parameter indicates the result of the transmission. This can take one of the values given in Table  
2785 52.

2786 Table 52 – Values of the *Result* parameter in *MAC\_DATA.confirm* primitive

| <i>Result</i>      | Description                 |
|--------------------|-----------------------------|
| <i>Success</i> = 0 | The send was successful.    |
| <i>Timeout</i> = 2 | The send process timed out. |

#### 2787 4.5.4.4 MAC\_DATA.indication

2788 The MAC\_DATA.indication primitive notifies the reception of data through a connection to the CL.

2789 The semantics of the primitive are as follows:

2790 *MAC\_DATA.indication*{*ConHandle*, *Data*, *DataLength*, *TimeReference*}

2791 The *ConHandle* parameter specifies the connection where the data was received. This handle is the one that  
2792 was obtained during the connection establishment primitives.

2793 The *Data* parameter is a buffer of octets that contains the CL data received through this connection.

2794 The *DataLength* parameter is the length of the *Data* parameter in octets.

2795 The *TimeReference* parameter is the time reference interchanged with the data. This *TimeReference*  
2796 parameter is optional; it is possible not receiving any time reference. From the primitive point of view the  
2797 act of not indicating a time reference will be considered a *NULL* time reference. The way to interchange this  
2798 parameter in the primitive not loosing precision and its absolute meaning are specific to the implementation.

### 2799 4.5.5 MAC Layer Management Entity SAPs

#### 2800 4.5.5.1 General

2801 The following primitives are all optional.

2802 The aim is to allow an external management entity to control Registration and Promotion of the Service  
2803 Node, demotion and Unregistration of a Service Node. The MAC layer would normally perform this  
2804 automatically; however, in some situations/applications it could be advantageous if this could be externally  
2805 controlled. Indications are also defined so that an external entity can monitor the status of the MAC.

#### 2806 4.5.5.2 MLME\_REGISTER

##### 2807 4.5.5.2.1 General

2808 The MLME\_REGISTER primitives are used to perform Registration and to indicate when Registration has been  
2809 performed.

#### 2810 4.5.5.2.2 MLME\_REGISTER.request

2811 The MLME\_REGISTER.request primitive is used to trigger the Registration process to a Subnetwork through  
2812 a specific Switch Node. This primitive may be used for enforcing the selection of a specific Switch Node that  
2813 may not necessarily be used if the selection is left automatic. The Base Node MLME function does not export  
2814 this primitive.

2815 The semantics of the primitive could be either of the following:

2816 *MLME\_REGISTER.request{ }*

2817 Invoking this primitive without any parameter simply invokes the Registration process in MAC and leaves the  
2818 selection of the Subnetwork and Switch Node to MAC algorithms. Using this primitive enables the MAC to  
2819 perform fully automatic Registration if such a mode is implemented in the MAC.

2820 *MLME\_REGISTER.request{SNA}*

2821 The SNA parameter specifies the Subnetwork to which Registration should be performed. Invoking the  
2822 primitive in this format commands the MAC to register only to the specified Subnetwork.

2823 *MLME\_REGISTER.request{SID}*

2824 The SID parameter is the SID (Switch Identifier) of the Switch Node through which Registration needs to be  
2825 performed. Invoking the primitive in this format commands the MAC to register only to the specified Switch  
2826 Node. In the case of a PLC+RF node, when SID = 0, the choice of the medium (PLC or RF) used to register is  
2827 taken by the lower layers and is left to individual implementations.

#### 2828 4.5.5.2.3 MLME\_REGISTER.confirm

2829 The MLME\_REGISTER.confirm primitive is used to confirm the status of completion of the Registration  
2830 process that was initiated by an earlier invocation of the corresponding request primitive. The Base Node  
2831 MLME function does not export this primitive.

2832 The semantics of the primitive are as follows:

2833 *MLME\_REGISTER.confirm{Result, SNA, SID}*

2834 The Result parameter indicates the result of the Registration. This can take one of the values given in  
2835 Table 53.

2836

2837 Table 53 – Values of the *Result* parameter in MLME\_REGISTER.confirm primitive

| <i>Result</i>      | Description                                                                       |
|--------------------|-----------------------------------------------------------------------------------|
| <i>Done</i> = 0    | Registration to specified SNA through specified Switch is completed successfully. |
| <i>Timeout</i> =2  | Registration request timed out .                                                  |
| <i>Rejected</i> =1 | Registration request is rejected by Base Node of specified SNA.                   |

| <b>Result</b>     | <b>Description</b>                                     |
|-------------------|--------------------------------------------------------|
| <i>NoSNA=8</i>    | Specified SNA is not within range.                     |
| <i>NoSwitch=9</i> | Switch Node with specified EUI-48 is not within range. |

2838

2839 The *SNA* parameter specifies the Subnetwork to which Registration is performed. This parameter is of  
 2840 significance only if *Result=0*.

2841 The *SID* parameter is the SID (Switch Identifier) of the Switch Node through which Registration is performed.  
 2842 This parameter is of significance only if *Result=0*.

#### 2843 **4.5.5.2.4 MLME\_REGISTER.indication**

2844 The MLME\_REGISTER.indication primitive is used to indicate a status change in the MAC. The Service Node  
 2845 is now registered to a Subnetwork.

2846 The semantics of the primitive are as follows:

2847 *MLME\_REGISTER.indication{SNA, SID}*

2848 The *SNA* parameter specifies the Subnetwork to which Registration is performed.

2849 The *SID* parameter is the SID (Switch Identifier) of the Switch Node through which Registration is performed.

#### 2850 **4.5.5.3 MLME\_UNREGISTER**

##### 2851 **4.5.5.3.1 General**

2852 The MLME\_UNREGISTER primitives are used to perform deregistration and to indicate when deregistration  
 2853 has been performed.

##### 2854 **4.5.5.3.2 MLME\_UNREGISTER.request**

2855 The MLME\_UNREGISTER.request primitive is used to trigger the Unregistration process. This primitive may  
 2856 be used by management entities if they require the Node to unregister for some reason (e.g. register through  
 2857 another Switch Node or to another Subnetwork). The Base Node MLME function does not export this  
 2858 primitive.

2859 The semantics of the primitive are as follows:

2860 *MLME\_UNREGISTER.request{}*

##### 2861 **4.5.5.3.3 MLME\_UNREGISTER.confirm**

2862 The MLME\_UNREGISTER.confirm primitive is used to confirm the status of completion of the unregister  
 2863 process initiated by an earlier invocation of the corresponding request primitive. The Base Node MLME  
 2864 function does not export this primitive.

2865 The semantics of the primitive are as follows:

2866 *MLME\_UNREGISTER.confirm{Result}*

2867 The *Result* parameter indicates the result of the Registration. This can take one of the values given in Table  
2868 54.

2869 **Table 54 – Values of the *Result* parameter in *MLME\_UNREGISTER.confirm* primitive**

| <b><i>Result</i></b> | <b>Description</b>                                                                           |
|----------------------|----------------------------------------------------------------------------------------------|
| <i>Done = 0</i>      | Unregister process completed successfully.                                                   |
| <i>Timeout =2</i>    | Unregister process timed out .                                                               |
| <i>Redundant=10</i>  | The Node is already in <i>Disconnected</i> functional state and does not need to unregister. |

2870

2871 On generation of *MLME\_UNREGISTER.confirm*, the MAC layer shall not perform any automatic actions that  
2872 may invoke the Registration process again. In such cases, it is up to the management entity to restart the  
2873 MAC functionality with appropriate *MLME\_REGISTER* primitives.

#### 2874 **4.5.5.3.4 *MLME\_UNREGISTER.indication***

2875 The *MLME\_UNREGISTER.indication* primitive is used to indicate a status change in the MAC. The Service Node  
2876 is no longer registered to a Subnetwork.

2877 The semantics of the primitive are as follows:

2878 *MLME\_UNREGISTER.indication{}*

#### 2879 **4.5.5.4 *MLME\_PROMOTE* and *MLME\_MP\_PROMOTE***

##### 2880 **4.5.5.4.1 General**

2881 The *MLME\_PROMOTE* primitives are used to perform promotion and to indicate when promotion has been  
2882 performed. They are used to trigger a promotion process in a Service Node (Terminal or Switch) in the  
2883 medium (PLC or RF) the node is connected to the network. They may also be used for the beacon modulation  
2884 change on the PLC medium.

2885 The *MLME\_MP\_PROMOTE* primitives have a similar scope and are only available in nodes that support Multi-  
2886 PHY promotion (*REG\_CAP.MP=1*). They are used to trigger a promotion process in a Service Node (Terminal  
2887 or Switch) in a medium (PLC or RF) different from the one the node is connected to the network. They may  
2888 also be used for the beacon modulation change on the PLC medium.

##### 2889 **4.5.5.4.2 *MLME\_PROMOTE.request***

2890 The *MLME\_PROMOTE.request* primitive is used to trigger the promotion process in a Service Node that is in  
2891 a *Terminal* functional state. Implementations may use such triggered promotions to optimize Subnetwork  
2892 topology from time to time. The value of *PRO.PNA* in the promotion message sent to the Base Node is  
2893 undefined and implementation-specific.

2894 The *MLME\_PROMOTE.request* primitive can also be used from a node that is already in a *Switch* state to ask  
2895 the BN for a Beacon PDU modulation change.

Base Node can use this primitive to ask a node to change its state from *Terminal* to *Switch* or, if the node is already in the *Switch* state, to adopt a new Beacon PDU modulation scheme.

The semantics of the primitive can be either of the following:

*MLME\_PROMOTE.request{}*

*MLME\_PROMOTE.request{BCN\_MODE}*

*MLME\_PROMOTE.request{EUI-48, BCN\_MODE}*

The EUI-48 parameter shall be used only by the Base Node to specify the address of the Node to which this promotion request shall be addressed. The MAC shall internally transfer this to an address used by the MAC layer.

The BCN\_MODE parameter specifies the Beacon PDU modulation scheme. If the primitive is called by a node in Switch state, this parameter indicates the requested Beacon PDU modulation scheme from the Switch node to the Base Node. If the primitive is called by the Base Node, this parameter indicates the modulation scheme that shall be communicated to the node during the promotion process or during the Beacon PDU modulation change process.

Allowed values for BCN\_MODE parameter are listed in Table 55.

Table 55 - Values of the BCN\_MODE parameter in MLME\_PROMOTE.request primitive.

| <b>BCN_MODE</b> | <b>Description</b>                                                                                    |
|-----------------|-------------------------------------------------------------------------------------------------------|
| DBPSK_F = 0     | BCN will be sent using DBPSK modulation with convolutional encoding enabled and robust mode disabled. |
| R_DBPSK = 1     | BCN will be sent using DBPSK modulation with robust mode enabled.                                     |
| R_DQPSK = 2     | BCN will be sent using DQPSK modulation with robust mode enabled.                                     |

#### 4.5.5.4.3 MLME\_PROMOTE.confirm

The MLME\_PROMOTE.confirm primitive is used to confirm the status of completion of a promotion process that was initiated by an earlier invocation of the corresponding request primitive.

The semantics of the primitive are as follows:

*MLME\_PROMOTE.confirm{Result}*

The *Result* parameter indicates the result of the Registration. This can take one of the values given in Table 56.

Table 56 – Values of the Result parameter in MLME\_PROMOTE.confirm primitive

| <b>Result</b>   | <b>Description</b>                                |
|-----------------|---------------------------------------------------|
| <i>Done</i> = 0 | Node is promoted to Switch function successfully. |

| <b>Result</b>             | <b>Description</b>                                     |
|---------------------------|--------------------------------------------------------|
| <i>Timeout =1</i>         | Promotion process timed out.                           |
| <i>Rejected=2</i>         | The Base Node rejected promotion request.              |
| <i>No Such Device = 4</i> | A device with the destination address cannot be found. |
| <i>Redundant=10</i>       | This device is already functioning as Switch Node.     |
| <i>OutofRange=12</i>      | Specified BCN_MODE is out of acceptable range.         |

2920

2921 In case an already promoted switch, which is requesting a Beacon PDU modulation change, receives an  
 2922 MLME\_PROMOTE.confirm{} rejecting the request, only the change request is supposed to be rejected, so the  
 2923 node shall continue sending the Beacon PDU as previously.

#### 2924 4.5.5.4.4 MLME\_PROMOTE.indication

2925 The MLME\_PROMOTE.indication primitive is used to indicate a status change in the MAC. The Service Node  
 2926 is now operating as a Switch. This primitive is not generated if a Beacon PDU modulation change occurs.

2927 The semantics of the primitive are as follows:

2928 *MLME\_PROMOTE.indication{}*

2929

#### 2930 4.5.5.4.5 MLME\_MP\_PROMOTE.request

2931 The MLME\_MP\_PROMOTE.request primitive is used to trigger a promotion process in a Service Node  
 2932 (Terminal or Switch) in a medium (PLC or RF) different from the one the node is connected to the network.  
 2933 Implementations may use such triggered promotions to optimize Subnetwork topology from time to time.  
 2934 The value of PRO.PNA in the promotion message sent to the Base Node is undefined and implementation-  
 2935 specific. The semantic used in this case is

2936 *MLME\_MP\_PROMOTE.request{PCH}*

2937 where, here and in the rest of this section, PCH is encoded in the same way as described in 4.4.2.6.5.1. For  
 2938 the different physical medium, every band (PLC) or channel (RF) available in the Service Node PLC Band Plan  
 2939 and RF band, respectively, can be used. This use of the primitive is associated to the process described in  
 2940 section 4.6.3.3.

2941 If a Service Node is connected to the network through the RF medium and is already a switch on the PLC  
 2942 medium (due to a successful Multi-PHY promotion, see 4.6.3.3), the MLME\_MP\_PROMOTE.request primitive  
 2943 can also be used to ask the BN for a Beacon PDU modulation change. The semantic used in this case is

2944 *MLME\_MP\_PROMOTE.request{BCN\_MODE}*

2945 where, here and in the rest of this section, the BCN\_MODE parameter follows the same rules described in  
 2946 4.5.5.4.2. This use of the primitive is associated to the process described in section 4.6.3.1.



2947 Base Node can use this primitive to ask a node (Terminal or Switch) to promote in a medium (PLC or RF)  
 2948 different from the one the node is connected to the network. The semantic used in this case is

2949 
$$\text{MLME\_MP\_PROMOTE.request}\{EUI48, BCN\_MODE, PCH\}$$

2950 where, here and in the rest of this section, the EUI48 parameter follows the same rules described in 4.5.5.4.2.  
 2951 For the different physical medium, every band (PLC) or channel (RF) available in the Service Node PLC Band  
 2952 Plan and RF band, respectively, can be used. This use of the primitive is associated to the process described  
 2953 in section 4.6.3.3.

2954 Base Node may also use this primitive to ask a Service Node that is connected to the network through the RF  
 2955 medium and is already a switch on the PLC medium to adopt a new Beacon PDU modulation scheme. The  
 2956 semantic used in this case is

2957 
$$\text{MLME\_MP\_PROMOTE.request}\{EUI48, BCN\_MODE\}$$

2958 This use of the primitive is associated to the process described in section 4.6.3.1.

#### 2959 **4.5.5.4.6 MLME\_MP\_PROMOTE.confirm**

2960 The MLME\_MP\_PROMOTE.confirm primitive is used to confirm the status of completion of a promotion  
 2961 process that was initiated by an earlier invocation of the corresponding request primitive.

2962 The semantics of the primitive are as follows:

2963 
$$\text{MLME\_MP\_PROMOTE.confirm}\{Result\}$$

2964 The *Result* parameter can take one of the values given in Table 56.

| <i>Result</i>             | <i>Description</i>                                                                                                                           |
|---------------------------|----------------------------------------------------------------------------------------------------------------------------------------------|
| <i>Done = 0</i>           | Node is promoted successfully.                                                                                                               |
| <i>Timeout = 1</i>        | Promotion process timed out.                                                                                                                 |
| <i>Rejected = 2</i>       | The Base Node rejected promotion request.                                                                                                    |
| <i>No Such Device = 4</i> | A device with the destination address cannot be found.                                                                                       |
| <i>Wrong Medium = 9</i>   | This device is connected to the network with the requested medium.<br>MLME_PROMOTE primitives shall be considered for this request.          |
| <i>Redundant = 10</i>     | This device is already functioning as Switch Node in the requested medium.                                                                   |
| <i>OutofRange = 12</i>    | Specified BCN_MODE is out of acceptable range or the band (PLC)/channel(RF) encoded in the PCH is out of the device's PLC band plan/RF band. |

2965

---

**2966 4.5.5.4.7 MLME\_MP\_PROMOTE.indication**

2967 The MLME\_MP\_PROMOTE.indication primitive is used to indicate a status change in the MAC. The Service  
2968 Node is operating as a Switch in a medium (PLC or RF) different from the one the node is connected to the  
2969 network. This primitive is not generated if a Beacon PDU modulation change occurs.

2970 The semantics of the primitive are as follows:

2971 *MLME\_MP\_PROMOTE.indication{PCH}*

2972 where PCH is encoded in the same way as described in 4.4.2.6.5.1.

**2973 4.5.5.5 MLME\_DEMOTE and MLME\_MP\_DEMOTE****2974 4.5.5.5.1 General**

2975 The MLME\_DEMOTE primitives are used to perform demotion and to indicate when demotion has been  
2976 performed. They are used to trigger a demotion process in a Switch in the medium (PLC or RF) the Switch is  
2977 connected to the network.

2978 The MLME\_MP\_DEMOTE primitives have a similar scope and are only available in nodes that support Multi-  
2979 PHY promotion (REG\_CAP.MP=1). They are used to trigger a demotion process in a Switch based upon the  
2980 LSIDs the Switch is using.

**2981 4.5.5.5.2 MLME\_DEMOTE.request**

2982 The MLME\_DEMOTE.request primitive is used to trigger a demotion process in a Service Node that is in a  
2983 *Switch* functional state. This primitive may be used by management entities to enforce demotion in cases  
2984 where the Node's default functionality does not automatically perform the process.

2985 The semantics of the primitive are as follows:

2986 *MLME\_DEMOTE.request{}*

**2987 4.5.5.5.3 MLME\_DEMOTE.confirm**

2988 The MLME\_DEMOTE.confirm primitive is used to confirm the status of completion of a demotion process  
2989 that was initiated by an earlier invocation of the corresponding request primitive.

2990 The semantics of the primitive are as follows:

2991 *MLME\_DEMOTE.confirm{Result}*

2992 The *Result* parameter indicates the result of the demotion. This can take one of the values given in Table 57.

2993 **Table 57 – Values of the *Result* parameter in MLME\_DEMOTE.confirm primitive**

| <b>Result</b>       | <b>Description</b>                                                                                                                                                                                                                                                                   |
|---------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>Done = 0</i>     | Node is demoted to Terminal function successfully. For nodes that support Multi-PHY promotion, node is demoted successfully and it is a Terminal or, in the case it was a Switch on two media, it is a Switch in one medium (different from the one it is connected to the network). |
| <i>Timeout =1</i>   | Demotion process timed out.                                                                                                                                                                                                                                                          |
| <i>Redundant=10</i> | This device is not a Switch on the medium it is connected to the network.                                                                                                                                                                                                            |

2994

2995 When a demotion has been triggered using the MLME\_DEMOTE.request, the Terminal will remain demoted.

#### 2996 **4.5.5.5.4 MLME\_DEMOTE.indication**

2997 The MLME\_DEMOTE.indication primitive is used to indicate a status change in the MAC. The Service Node is  
 2998 now operating as a Terminal. For nodes that support Multi-PHY promotion, it may also indicate that the  
 2999 Service Node previously acting as a Switch on two media is now a Switch only in one medium (different from  
 3000 the one it is connected to the network).

3001 The semantics of the primitive are as follows:

3002 *MLME\_DEMOTE.indication{}*

#### 3003 **4.5.5.5.5 MLME\_MP\_DEMOTE.request**

3004 The MLME\_MP\_DEMOTE.request primitive is used to trigger a demotion process in a Service Node that is in  
 3005 a *Switch* functional state. This primitive may be used by management entities to enforce demotion in cases  
 3006 where the Node's default functionality does not automatically perform the process.

3007 The semantics of the primitive are as follows:

3008 *MLME\_MP\_DEMOTE.request{LSID}*

3009 Here, LSID is a Local Switch Identifier. Note that, as the Service Node may be Switch on two media (PLC and  
 3010 RF), it may have two LSIDs (see 4.6.4).

#### 3011 **4.5.5.5.6 MLME\_MP\_DEMOTE.confirm**

3012 The MLME\_MP\_DEMOTE.confirm primitive is used to confirm the status of completion of a demotion process  
 3013 that was initiated by an earlier invocation of the corresponding request primitive.

3014 The semantics of the primitive are as follows:

3015 *MLME\_MP\_DEMOTE.confirm{Result}*

3016 The *Result* parameter indicates the result of the demotion. This can take one of the values given in Table 58.

3017 **Table 58 - Values of the *Result* parameter in MLME\_MP\_DEMOTE.confirm primitive**

| <b>Result</b>          | <b>Description</b>                                                                                                           |
|------------------------|------------------------------------------------------------------------------------------------------------------------------|
| <i>Done = 0</i>        | Node is demoted successfully. Node is a Terminal or, in the case it was a Switch on two media, it is a Switch in one medium. |
| <i>Timeout = 1</i>     | Demotion process timed out.                                                                                                  |
| <i>Wrong LSID = 15</i> | This device is not a Switch with this LSID.                                                                                  |

#### 3018 4.5.5.5.7 MLME\_MP\_DEMOTE.indication

3019 The MLME\_MP\_DEMOTE.indication primitive is used to indicate a status change in the MAC. The Service  
 3020 Node is now operating as a Terminal. It may also indicate that the Service Node previously acting as a Switch  
 3021 on two media is now a Switch only in one medium.

3022 The semantics of the primitive are as follows:

3023 
$$MLME\_MP\_DEMOTE.indication\{LSID\}$$

3024 where LSID is the demoted Local Switch Identifier.

#### 3025 4.5.5.6 MLME\_RESET

##### 3026 4.5.5.6.1 General

3027 The MLME\_RESET primitives are used to reset the MAC into a known good status.

##### 3028 4.5.5.6.2 MLME\_RESET.request

3029 The MLME\_RESET.request primitive results in the flushing of all transmit and receive buffers and the resetting  
 3030 of all state variables. As a result of invoking of this primitive, a Service Node will transit from its present  
 3031 functional state to the *Disconnected* functional state.

3032 The semantics of the primitive are as follows:

3033 
$$MLME\_RESET.request\{\}$$

##### 3034 4.5.5.6.3 MLME\_RESET.confirm

3035 The MLME\_RESET.confirm primitive is used to confirm the status of completion of a reset process that was  
 3036 initiated by an earlier invocation of the corresponding request primitive. On the successful completion of the  
 3037 reset process, the MAC entity shall restart all functions starting from the search for a Subnetwork (4.3.1).

3038 The semantics of the primitive are as follows:

3039 
$$MLME\_RESET.confirm\{Result\}$$

3040 The *Result* parameter indicates the result of the reset. This can take one of the values given below.

3041 **Table 59 – Values of the *Result* parameter in MLME\_RESET.confirm primitive**

| <b>Result</b>    | <b>Description</b>                                       |
|------------------|----------------------------------------------------------|
| <i>Done = 0</i>  | MAC reset completed successfully.                        |
| <i>Failed =1</i> | MAC reset failed due to internal implementation reasons. |

#### 3042 4.5.5.7 MLME\_GET

##### 3043 4.5.5.7.1 General

3044 The MLME\_GET primitives are used to retrieve individual values from the MAC, such as statistics.

##### 3045 4.5.5.7.2 MLME\_GET.request

3046 The MLME\_GET.request queries information about a given PIB attribute.

3047 The semantics of the primitive are as follows:

3048 *MLME\_GET.request{PIBAttribute}*

3049 The *PIBAttribute* parameter identifies specific attributes as listed in the *Id* fields of tables that list PIB  
3050 attributes (Section 6.2.3).

##### 3051 4.5.5.7.3 MLME\_GET.confirm

3052 The MLME\_GET.confirm primitive is generated in response to the corresponding MLME\_GET.request  
3053 primitive.

3054 The semantics of this primitive are as follows:

3055 *MLME\_GET.confirm{status, PIBAttribute, PIBAttributeValue}*

3056 The *status* parameter reports the result of requested information and can have one of the values given in  
3057 Table 60.

3058 **Table 60 – Values of the *status* parameter in MLME\_GET.confirm primitive**

| <b>Result</b>     | <b>Description</b>                                            |
|-------------------|---------------------------------------------------------------|
| <i>Done = 0</i>   | Parameter read successfully.                                  |
| <i>Failed =1</i>  | Parameter read failed due to internal implementation reasons. |
| <i>BadAttr=11</i> | Specified <i>PIBAttribute</i> is not supported.               |

3059

3060 The *PIBAttribute* parameter identifies specific attributes as listed in *Id* fields of tables that list PIB attributes  
3061 (Section 6.2.3.5).

3062 The *PIBAtributeValue* parameter specifies the value associated with a given *PIBAttribute*

#### 3063 4.5.5.8 MLME\_LIST\_GET

##### 3064 4.5.5.8.1 General

3065 The MLME\_LIST\_GET primitives are used to retrieve a list of values from the MAC.

##### 3066 4.5.5.8.2 MLME\_LIST\_GET.request

3067 The MLME\_LIST\_GET.request queries for a list of values pertaining to a specific class. These special classes of  
3068 PIB attributes are listed in Table 110.

3069 The semantics of the primitive are as follows:

3070 *MLME\_LIST\_GET.request{PIBListAttribute}*

3071 The *PIBListAttribute* parameter identifies a specific list that is requested by the management entity. The  
3072 possible values of *PIBListAttribute* are listed in 6.2.3.5.

##### 3073 4.5.5.8.3 MLME\_LIST\_GET.confirm

3074 The MLME\_LIST\_GET.confirm primitive is generated in response to the corresponding  
3075 MLME\_LIST\_GET.request primitive.

3076 The semantics of this primitive are as follows:

3077 *MLME\_LIST\_GET.confirm{status, PIBListAttribute, PIBListAttributeValue}*

3078 The *status* parameter reports the result of requested information and can have one of the values given in  
3079 Table 61

3080 Table 61 – Values of the *status* parameter in MLME\_LIST\_GET.confirm primitive

| <b>Result</b>     | <b>Description</b>                                            |
|-------------------|---------------------------------------------------------------|
| <i>Done = 0</i>   | Parameter read successfully.                                  |
| <i>Failed =1</i>  | Parameter read failed due to internal implementation reasons. |
| <i>BadAttr=11</i> | Specified <i>PIBListAttribute</i> is not supported.           |

3081

3082 The *PIBListAttribute* parameter identifies a specific list as listed in the *Id* field of Table 110.

3083 The *PIBListAttributeValue* parameter contains the actual listing associated with a given *PIBListAttribute*

#### 3084 4.5.5.9 MLME\_SET

##### 3085 4.5.5.9.1 General

3086 The MLME\_SET primitives are used to set configuration values in the MAC.

##### 3087 4.5.5.9.2 MLME\_SET.request

3088 The MLME\_SET.request requests to configure a given PIB attribute.

3089 The semantics of the primitive are as follows:

3090 *MLME\_SET.request{PIBAttribute, PIBAttributeValue}*

3091 The *PIBAttribute* parameter identifies a specific attribute as listed in the *Id* fields of tables that list PIB  
3092 attributes (Section 6.2.3).

3093 The *PIBAttributeValue* parameter specifies the value associated with given *PIBAttribute*.

#### 3094 4.5.5.9.3 MLME\_SET.confirm

3095 The MLME\_SET.confirm primitive is generated in response to the corresponding MLME\_SET.request  
3096 primitive.

3097 The semantics of this primitive are as follows:

3098 *MLME\_SET.confirm{result}*

3099 The *result* parameter reports the result of requested information and can have one of the values given in  
3100 Table 62.

3101 Table 62 – Values of the *Result* parameter in MLME\_SET.confirm primitive

| <b>Result</b>        | <b>Description</b>                                             |
|----------------------|----------------------------------------------------------------|
| <i>Done = 0</i>      | Given value successfully set for specified attribute.          |
| <i>Failed =1</i>     | Failed to set the given value for specified attribute.         |
| <i>BadAttr=11</i>    | Specified <i>PIBAttribute</i> is not supported.                |
| <i>OutOfRange=12</i> | Specified <i>PIBAttributeValue</i> is out of acceptable range. |
| <i>ReadOnly=13</i>   | Specified <i>PIBAttribute</i> is read only.                    |

3102

3103 The *PIBAttribute* parameter identifies a specific attribute as listed in the *Id* fields of tables that list PIB  
3104 attributes (Section 6.2.3).

3105 The *PIBAttributeValue* parameter specifies the value associated with a given *PIBAttribute*.

## 3106 4.6 MAC procedures

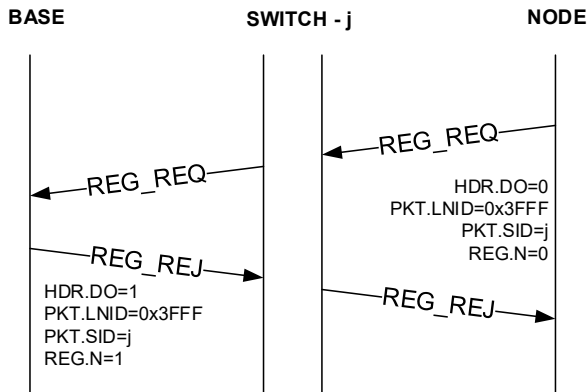
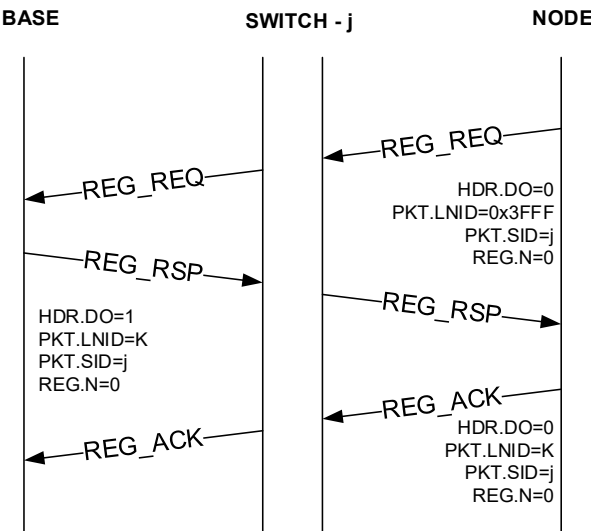
### 3107 4.6.1 Registration process

#### 3108 4.6.1.1 General

3109 The initial Service Node start-up (4.3.1) is followed by a Registration process. A Service Node in a  
3110 *Disconnected* functional state shall transmit a REG control packet to the Base Node in order to get itself  
3111 included in the Subnetwork. Since no LNID or SID is allocated to a Service Node at this stage, the PKT.LNID  
3112 field shall be set to all 1s and the PKT.SID field shall contain the SID of the Switch Node through which it seeks  
3113 attachment to the Subnetwork.

- 3114 Base Nodes may use a Registration request as an authentication mechanism. However this specification does  
3115 not recommend or forbid any specific authentication mechanism and leaves this choice to implementations.
- 3116 For all successfully accepted Registration requests, the Base Node shall allocate an LNID that is unique within  
3117 the domain of the Switch Node through which the attachment is realized. This LNID shall be indicated in the  
3118 PKT.LNID field of response (REG\_RSP). The assigned LNID, in combination with the SID of the Switch Node  
3119 through which the Service Node is registered, would form the NID of the registering Node.
- 3120 Registration is a three-way process. The Base Node answers to the REG\_REQ - registration request - sent by  
3121 a Service Node by means of a REG\_RSP message, which shall be acknowledged by the Service Node with a  
3122 REG\_ACK message.
- 3123 Service Nodes report their capabilities to the Base Node during registration (REG\_REQ), as specified in  
3124 4.4.2.6.3. On top of that, a Base Node is able to configure some parameters in Service Nodes when answering  
3125 (REG\_RSP) to a registration request.
- 3126 • Dynamic robustness-management is enabled by default. Nonetheless, the Base Node may disable  
3127 dynamic robustness-management and fix a specific modulation scheme, thus not allowing Service  
3128 Node(s) to dynamically switch to a different modulation scheme.  
3129
- 3130 The configured value is stored by the Service Node as *“macRobustnessManagement”*.  
3131
- 3132 • *Segmentation And Reassembly (SAR)* packet size: The packet size used by Convergence Layer’s SAR  
3133 Service is not configured by the Base Node by default. Nonetheless, a Base Node may fix a specific  
3134 SAR packet size if required. For more information about Convergence Layer’s SAR Service, please  
3135 refer to section 5.6.2.1.3
- 3136 The configured value is stored by the Service Node as *“macSARsize”*.
- 3137 Configuration of the two parameters mentioned above during registration provides a static network  
3138 configuration. This configuration can be changed by the Base Node, either starting a new registration process  
3139 or setting the corresponding Service Node’s PIB variables remotely.
- 3140 Figure 79 represents a successful Registration process and Figure 80 shows a Registration request that is  
3141 rejected by the Base Node. Details on specific fields that distinguish one Registration message from the other  
3142 are given in Table 22. The registration process security-related steps are explained in Section 4.6.1.2.
- 3143 The REG control packet, in all its usage variants, is transmitted unencrypted, but specified fields (REG.SWK  
3144 and REG.WK) are encrypted with context-specific encryption keys as explained in Section 4.4.2.6.3. The  
3145 encryption of REG.WK in REG\_RSP, its decryption at the receiving end and subsequent encrypted  
3146 retransmission using a different encryption key authenticates that the REG\_ACK is from the intended  
3147 destination.





When assigning an LNID, the Base Node shall not reuse an LNID released by an unregister process before ( $macCtrlMsgFailTime + macMinCtReTxTimer$ ) seconds, to ensure that all retransmitted packets have left the Subnetwork. Similarly, the Base Node shall not reuse an LNID released by the Keep-Alive process before  $T_{keep\_alive}$  seconds, using the last known acknowledged  $T_{keep\_alive}$  value, or if larger, the last unacknowledged  $T_{keep\_alive}$ , for the Service Node using the LNID. When security is being used in the network, the Base Node shall not reuse a LNID without first changing the Subnetwork Working Key.

During network startup where the whole network is powered on at once, there will be considerable contention for the medium. It is recommended to add randomness to the first REG\_REQ transmission, as well as to all subsequent retransmissions. It is recommended to wait a random delay before the first REG\_REQ message. This delay should be in range from 0 to at least 10% of  $macCtrlMsgFailTime$ . Similarly a random delay may be added to each retransmission.

#### 4.6.1.2 Security registration process

Figure 79 represents the registration process. When security profile 1 or 2 is utilized, additional action is required by the Base and Terminal Nodes to ensure successful registration.

1. The Terminal Node generates a challenge (see Section 4.3.8.2.2.3).
2. The challenge is included in the REG\_REQ and the REG\_REQ is authenticated with REGK.

- 3. The Base Node validates that REG\_REQ is properly authenticated.
- 4. The SWK and WK are key wrapped with KWK. The REG\_RSP is authenticated with REGK and the Terminal Node challenge is concatenated.
- 5. The Terminal Node validates that REG\_RSP is properly authenticated, including the concatenated challenge
- 6. The Terminal Node updates WK and SWK.
- 7. The REG\_ACK is authenticated with WK. The first Nonce is required for AES-CCM (Set to 0, then counted up for every packet.)
- 8. The Base Node validates REG\_ACK. The registration is invalidated on error

4.6.2 Unregistration process

At any point in time, either the Base Node or the Service Node may decide to close an existing registration. This version of the specification does not provide provision for rejecting an unregistration request. The Service Node or Base Node that receives an unregistration request shall acknowledge its receipt and take appropriate actions.

Following a successful unregistration, a Service Node shall move back from its present functional state to a *Disconnected* functional state and the Base Node may re-use any resources that were reserved for the unregistered Node.

Figure 81 shows a successful unregistration process initiated by a Service Node and Figure 82 shows an unregistration process initiated by the Base Node. Details on specific fields that identify unregistration requests in REG control packets are given in Table 23.

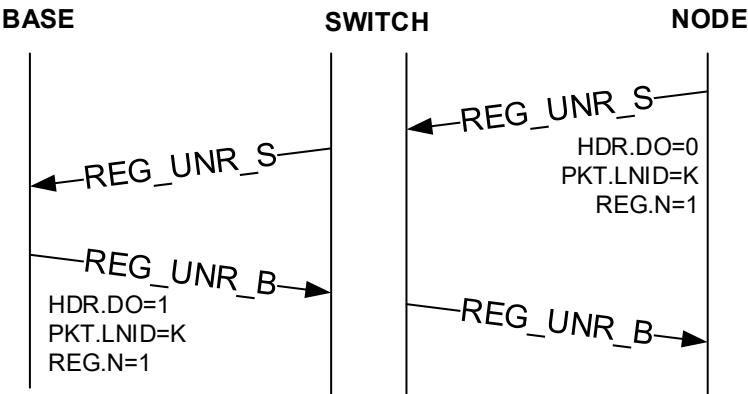


Figure 81 – Unregistration process initiated by a Terminal Node

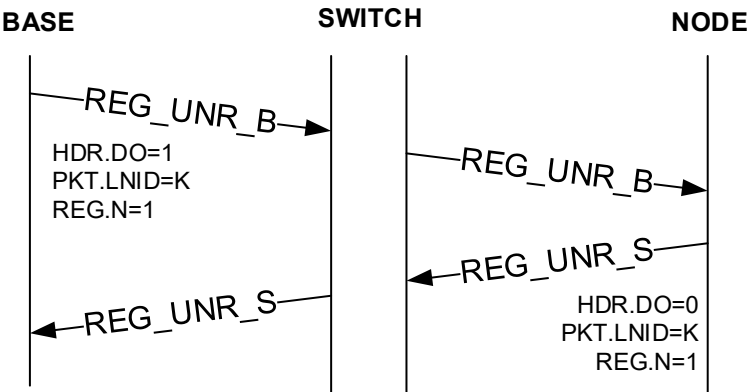


Figure 82 – Unregistration process initiated by the Base Node

### 4.6.3 Promotion process

The promotion process described in this section occurs on the same medium and on the same band (PLC) or channel (RF) a Service Node is connected to the network. The process, as well as the related messages (e.g. see 4.4.2.6.5) and primitives (e.g. see 4.5.5.4.2) are described for a Terminal becoming Switch. For nodes that support Multi-PHY promotion (see 4.6.3.3), this process may also apply to a Service Node that it is already a Switch on a medium different from the one it is connected to the network.

A Service Node that does not receive any BPDUs may transmit PNPDU. Any Terminal Node receiving PNPDU may generate a promotion request towards Base Node, which upon acceptance from Base Node, will result in transition of the requesting Terminal Node to Switch and therefore scale the Subnetwork to facilitate PNPDU transmitting Service Node to join.

Note: A Subnetwork that operates in backward compatibility-mode as enumerated in 4.8, shall silently discard PNPDU that indicate lack of support for backward compatibility-mode i.e. PNH.VER = 1 and PNH.CAP\_BC = 0.

The Base Node examines promotion requests during a period of time. It may use the address of the new Terminal, provided in the promotion-request packet, to decide whether or not to accept the promotion. It decides which Service Node shall be promoted, if any, sending a promotion response. The other Nodes do not receive any response to their promotion request to avoid Subnetwork saturation. Eventually, the Base Node may send a rejection if any special situation occurs. If the Subnetwork is specially preconfigured, the Base Node may send Terminal Node promotion requests directly to a Terminal Node.

When a Terminal Node requests promotion, the PRO.NSID field in the PRO\_REQ\_S message shall be set to all 1s. The PRO.NSID field shall contain an LSID allocated to the promoted Node in the PRO\_REQ\_B message. The acknowledging Switch Node shall set the PRO.NSID field in its PRO\_ACK to the newly allocated LSID. This final PRO\_ACK shall be used by intermediate Switch Nodes to update their switching tables.

When reusing LSIDs that have been released by a demotion process, the Base Node shall not allocate the LSID until after  $(macCtrlMsgFailTime + macMinCtReTxTimer)$  seconds to ensure all retransmit packets that might use that LSID have left the Subnetwork.

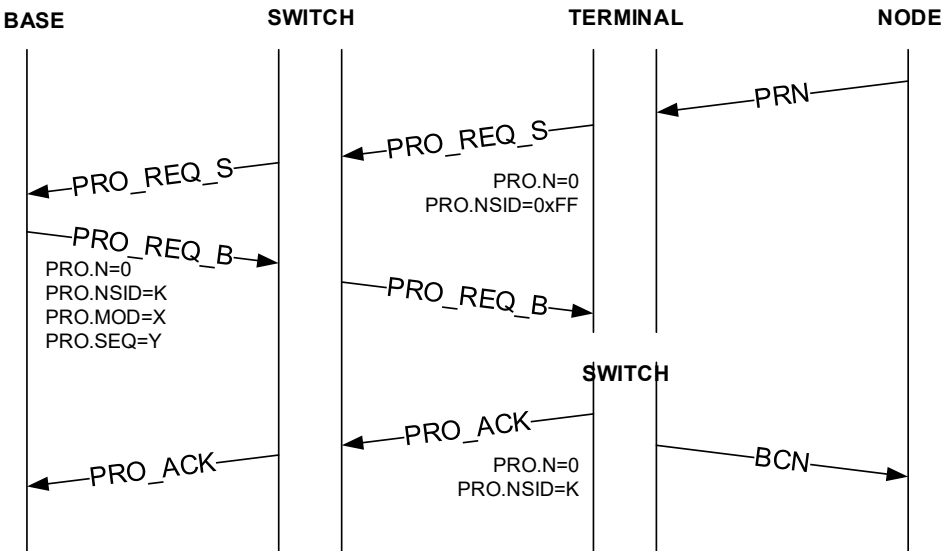


Figure 83 – Promotion process initiated by a Service Node

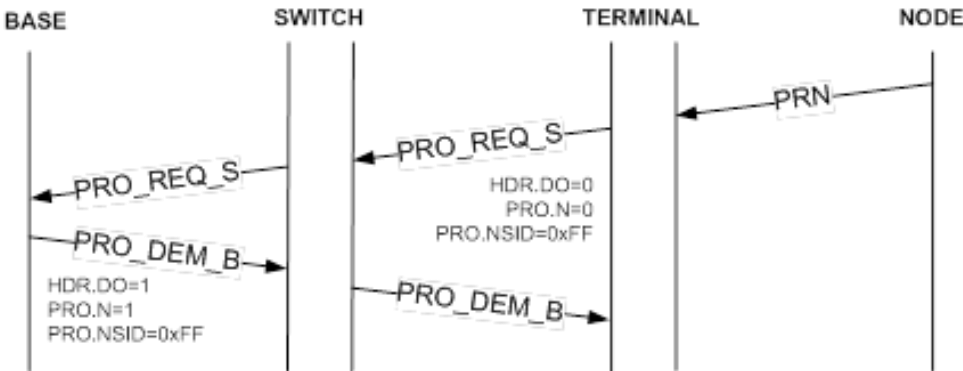


Figure 84 – Promotion process rejected by the Base Node

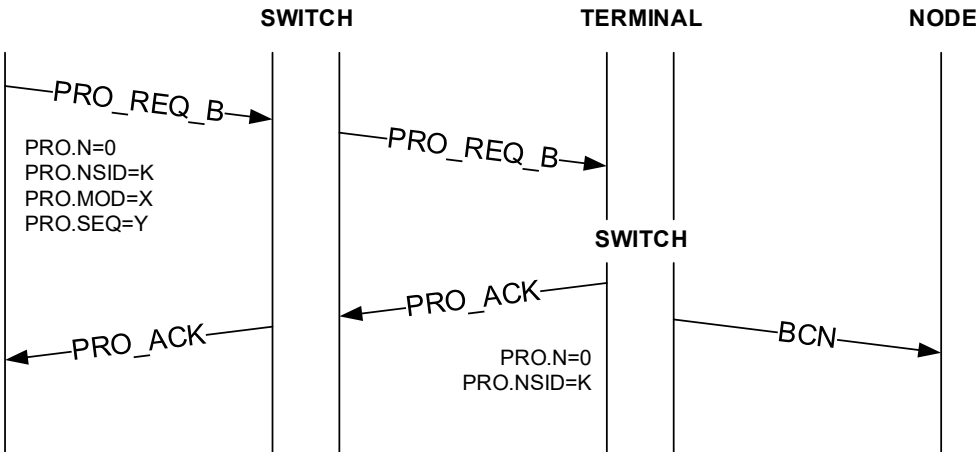


Figure 85 – Promotion process initiated by the Base Node

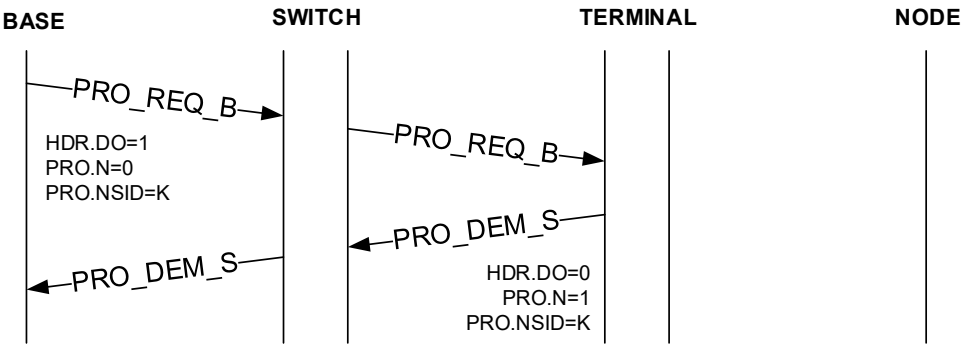


Figure 86 – Promotion process rejected by a Service Node

4.6.3.1 BPDU modulation change

In the PLC medium, it is possible, for the Switch Node, to change modulation scheme used to send BPDUs. In order to do so a new PRO\_REQ\_S message is sent with an indication of the new desired modulation scheme to be used in PRO.MOD field. On reception of this PRO\_REQ\_S the Base Node shall send a PRO\_ACK packet to accept the change request, or send a PRO\_NACK packet to reject the change request. The Base Node can not indicate a new BPDU modulation scheme in the PRO.MOD field that is different from the requested one. In such a case the Base Node can accept the requested modulation and initiate another beacon modulation change by itself. The Switch would then either acknowledge the reception by sending a PRO\_ACK (accept the new modulation) or a PRO\_NACK packet (reject the new modulation). In case an explicit denial is issued by the Base Node, the Switch shall keep on sending the Beacon PDUs without changing to the new modulation scheme. Switch Node shall not sent PRO\_ACK packet in case of explicit reject.

The Beacon PDU modulation change process can also be initialized by the Base Node. In this case the Switch Node shall send a PRO\_ACK packet if it can perform the Beacon PDU modulation change otherwise it shall send a PRO\_NACK.

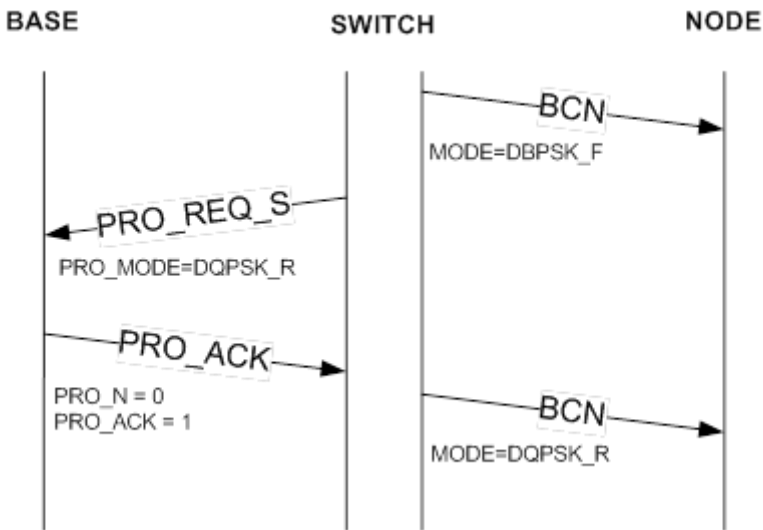


Figure 87 – BCN modulation change request initiated by the Switch.

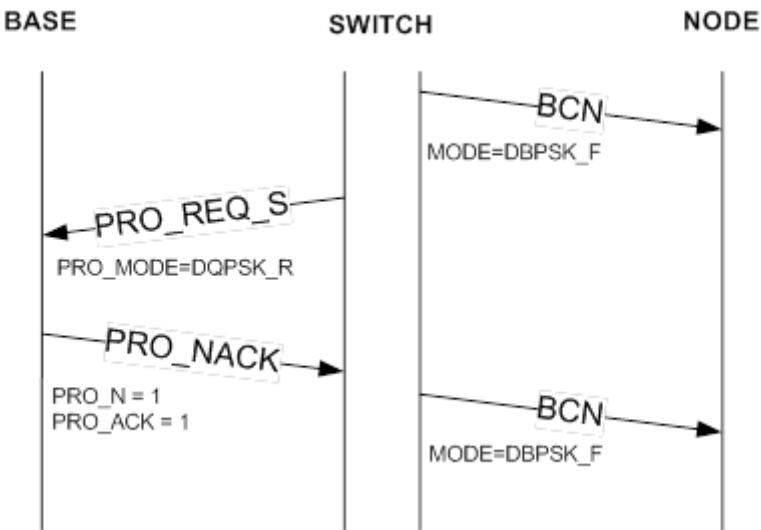


Figure 88 – BCN modulation change request initiated by the Switch and rejected by the Base Node.

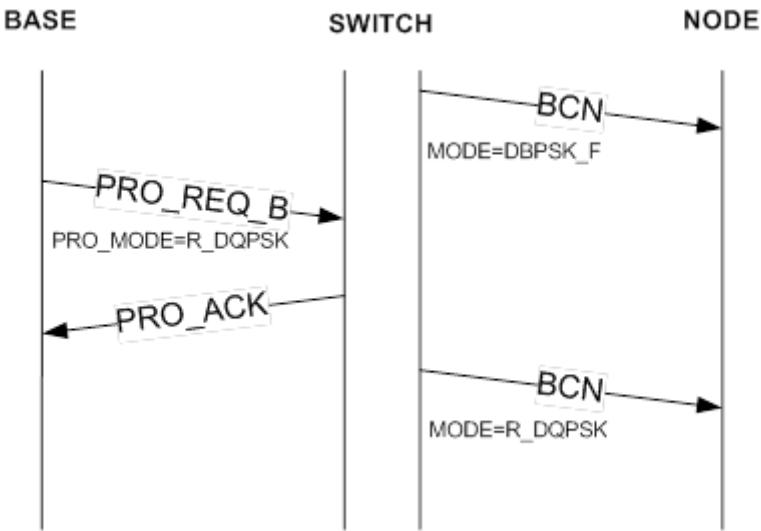


Figure 89 - BCN modulation change request initiated by the Base Node.

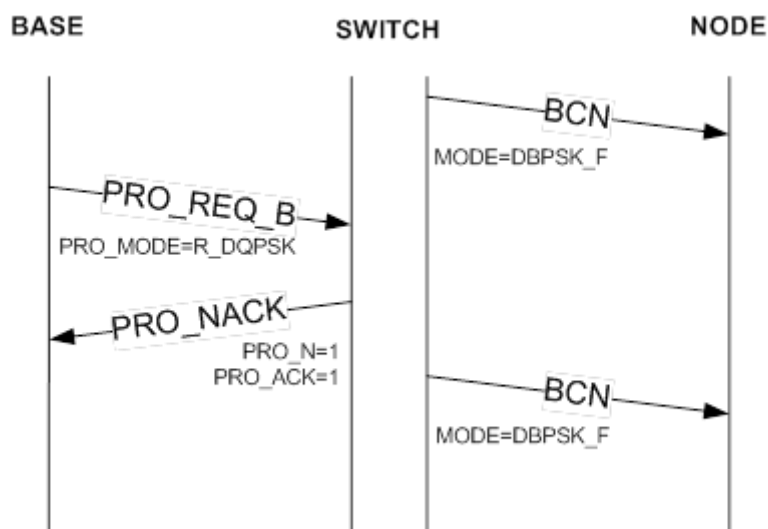


Figure 90 - BCN modulation change request initiated by the Base Node and rejected by the Switch

During a Promotion procedure the Base Node assigns resources to new Switch Node in order to transmit its BPDUs. These changes shall take effect only on a super-frame boundary. In case these changes require a change in frame structure, the Base Node shall send a FRA packet to inform the entire network.

#### 4.6.3.2 Double switching procedure

This section has been written considering that the support of robust modes is optional for PLC-only devices. For PLC+RF devices, the support of robust modes is mandatory (these devices indicate REG.CAP\_R=1, PRO.PN\_R=1, PNH.CAP\_R=1, BCN.CAP\_R=1). Moreover, more generally, PLC+RF devices shall be part of networks where robust modes are supported on the PLC medium (these networks may also include PLC-only devices supporting robust modes).

With the previous considerations, the procedure described in what follows, named double switching, is not supported by PLC+RF devices (and, obviously, by RF devices) and will be described for PLC-only devices in the context of PLC-only Subnetworks.

Certain PLC-only Subnetworks may have a mix of device-types between ones that can support Type A PHY frames only and ones that support both Type A and Type B PHY frames. In such cases, a Switch Node that acts as switching point for both kinds of devices, may need to transmit BPDUs using both types of PHY frames.

In order to be able to transmit BPDUs using both types of PHY frames, a Switch Node needs to undergo a second promotion procedure. The first promotion is carried out in the usual manner as enumerated 4.6.3. When a Switch Node identifies need to transmit its BPDUs in additional modulation scheme, it starts a second promotion procedure. The Switch Node uses PRO packets with the PRO.DS bit set to one. Additionally, the PRO\_REQ\_S packet shall fill LSID of the requesting Switch Node in PRO.SID field.

When a Switch Node has two BPDUs to send it may ask to change modulation scheme only for the robust beacon, passing from the DBPSK\_R to DQPSK\_R or viceversa following procedure enumerated in 4.6.3.1.

3277 To stop sending one type of BPDUs, the demotion procedure is used. In this case the PRO.MOD field indicates  
 3278 which beacon shall not be transmit and the PRO.DS field set to 1 idicates that the node is asking to stop  
 3279 sending one type of beacon. If the PRO.DS field is set to 0 the node is asking for a full demotion, stop sending  
 3280 both BPDUs and transit back to *Terminal* funcional state.

3281 By having possibility to provide connectivity for Type A only devices and for devices that require Type B  
 3282 frames, the Switch Node shall guarantee delivery of multicast and broadcast packets. In simplified  
 3283 implementations, the Switch Node can transmit these types of packets twice, one with the Type A frame and  
 3284 one with the Type B. Multicast data shall be switched in conformance to procedures enumerated in 4.6.7.4.3.

3285 For broadcast data, the Switch Node shall start to send packets twice after it succesfully performs the double  
 3286 beacon slot allocation, and it shall stop sending one of the two type of packets when the demotion procedure  
 3287 is completed for that specific type of frame.

3288 Devices that are able to understand both frames, Type A or Type B may receive same data twice, at each  
 3289 modulation scheme. Such devices shall be intelligent enough to discard the duplicate receipt of data..

#### 3290 **4.6.3.3 Extension for Multi-PHY promotion procedure**

3291 When the base node notifies through the registration process that the multiPHY promotion is supported:  
 3292 REG.CAP\_MP=1, if also supported by a service node, the latter can start a promotion process in a different  
 3293 physical medium (PLC or RF) than the one through which it is connected to the network. For the different  
 3294 physical medium, it can select every band (PLC) or channel (RF) available in its PLC Band Plan and RF band,  
 3295 respectively.

3296 If a node (terminal or switch) can hear PNPDU's in a different medium (PLC or RF), it may send a new  
 3297 promotion request to be switch in other medium, to do that, It must use the PRO\_REQ\_S\_MultiPHY message.

3298 The physical medium and the band (PLC) or channel (RF) will be coded in the PRO.PCH field of the message.  
 3299 When the network is configured to use channel hopping (see 4.6.11) on the RF PHY medium, the PRO.PCH  
 3300 field has a fixed coding with all bits equal to 1.

3301

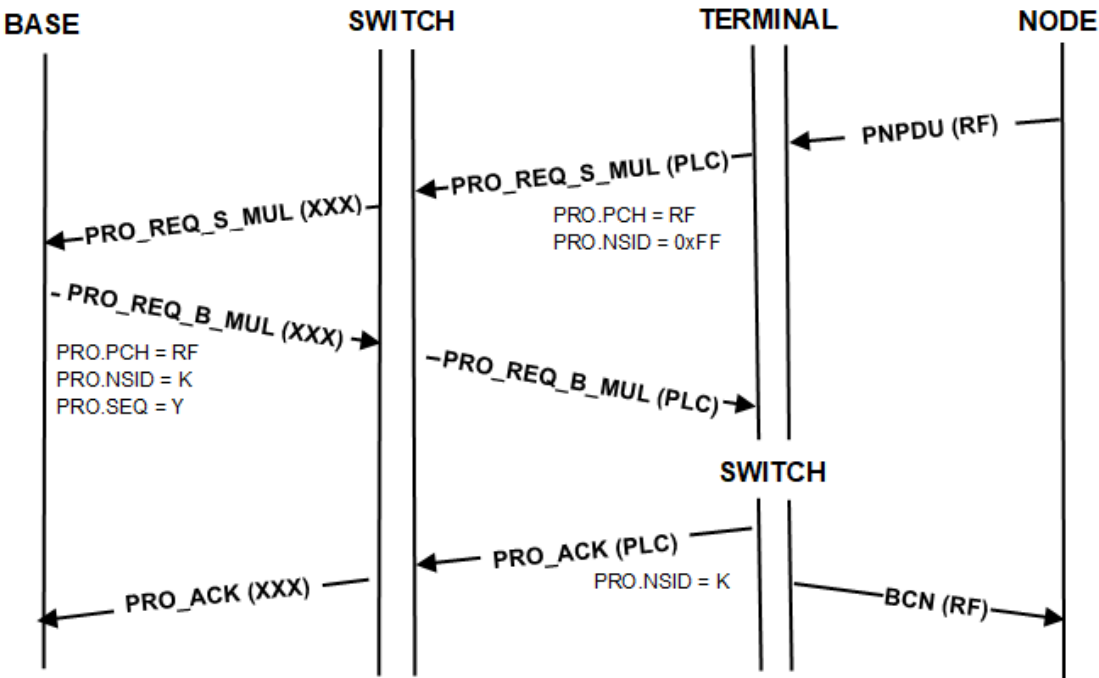
3302 The PRO.NSID field in the PRO\_REQ\_S\_MultiPHY message shall be set to all 1s. If the Base Node accepts a  
 3303 promotion request, it allocates an LSID to the requesting node (terminal or switch) using the field PRO.NSID  
 3304 of the PRO\_REQ\_B\_MultiPHY message. This LSID shall be different from the one possibly already allocated  
 3305 through PRO\_REQ\_B using the promotion process described in 4.6.3, so that a switch could have two LSIDs.  
 3306 The physical medium coded in the PRO.PCH field of PRO\_REQ\_B\_MultiPHY message shall be the same as the  
 3307 one coded in the PRO\_REQ\_S\_MultiPHY message (e.g. the Base Node shall not accept a promotion request  
 3308 on the PLC medium with an RF PRO.PCH field in the PRO\_REQ\_B\_MultiPHY).

3309 If the Base Node is specially configured, it may send a PRO\_REQ\_B\_MultiPHY message to a node (terminal or  
 3310 switch) to promote it in a physical medium (PLC or RF) different from the one through which the node is  
 3311 connected to the network. For the different physical medium, it can select every band (PLC) or channel (RF)  
 3312 available in its PLC Band Plan and RF band, respectively.

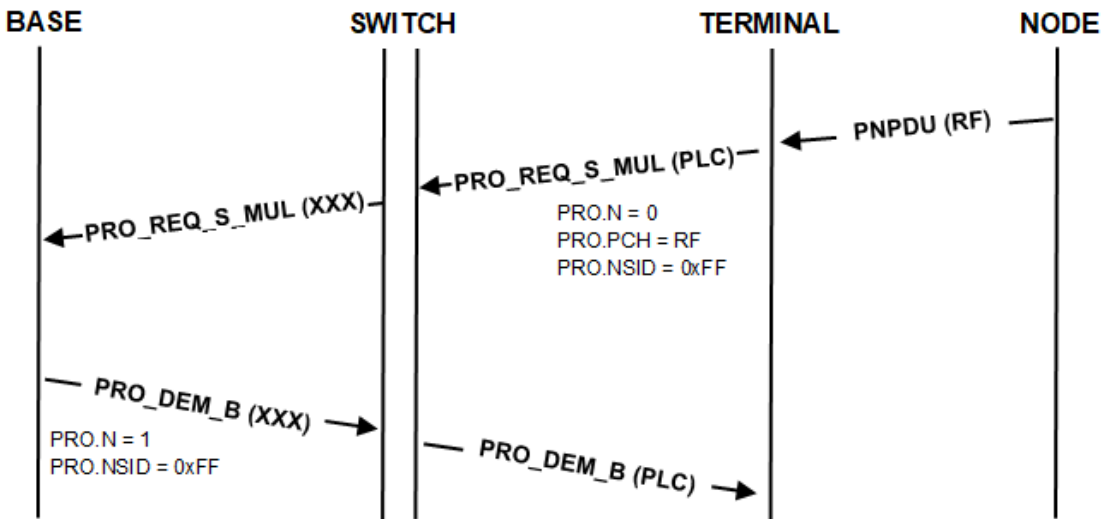


3313 With this method, all the normal operations (promotion acknowledgement/not acknowledgement,  
3314 demotion, change of modulation) could be done with the PRO\_REQ\_S and PRO\_REQ\_B messages using the  
3315 LSID allocated through PRO\_REQ\_B\_MultiPHY message.

3316 A switch shall use on the same medium (i.e. PLC or RF) 1 band (PLC) and/or 1 channel (RF). The use of multiple  
3317 RF channels is admitted under the channel hopping procedure.



3318  
3319 Figure 91 – Multi-PHY promotion process initiated by Service Node with the most relevant values  
3320



3321  
3322 Figure 92 - Multi-PHY promotion process rejected by Base Node with the most relevant values  
3323

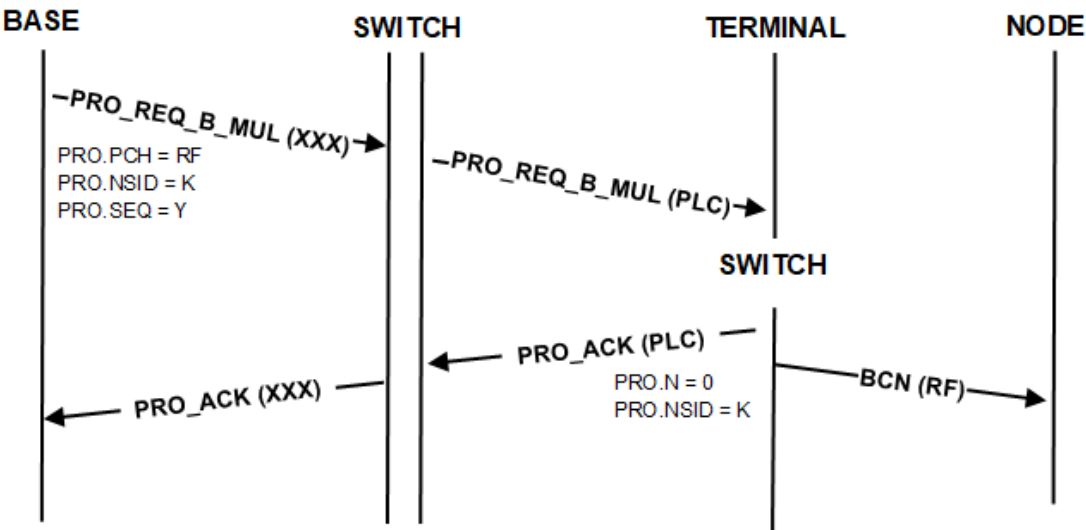


Figure 93 – Multi-PHY promotion process initiated by Base Node with the most relevant values

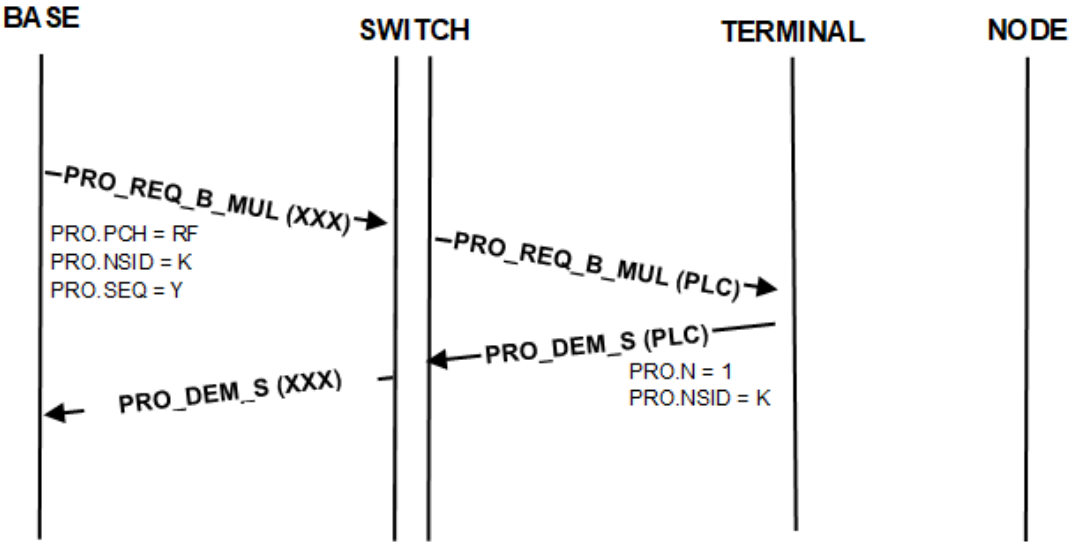


Figure 94 – Multi-PHY promotion process rejected by Service Node with the most relevant values

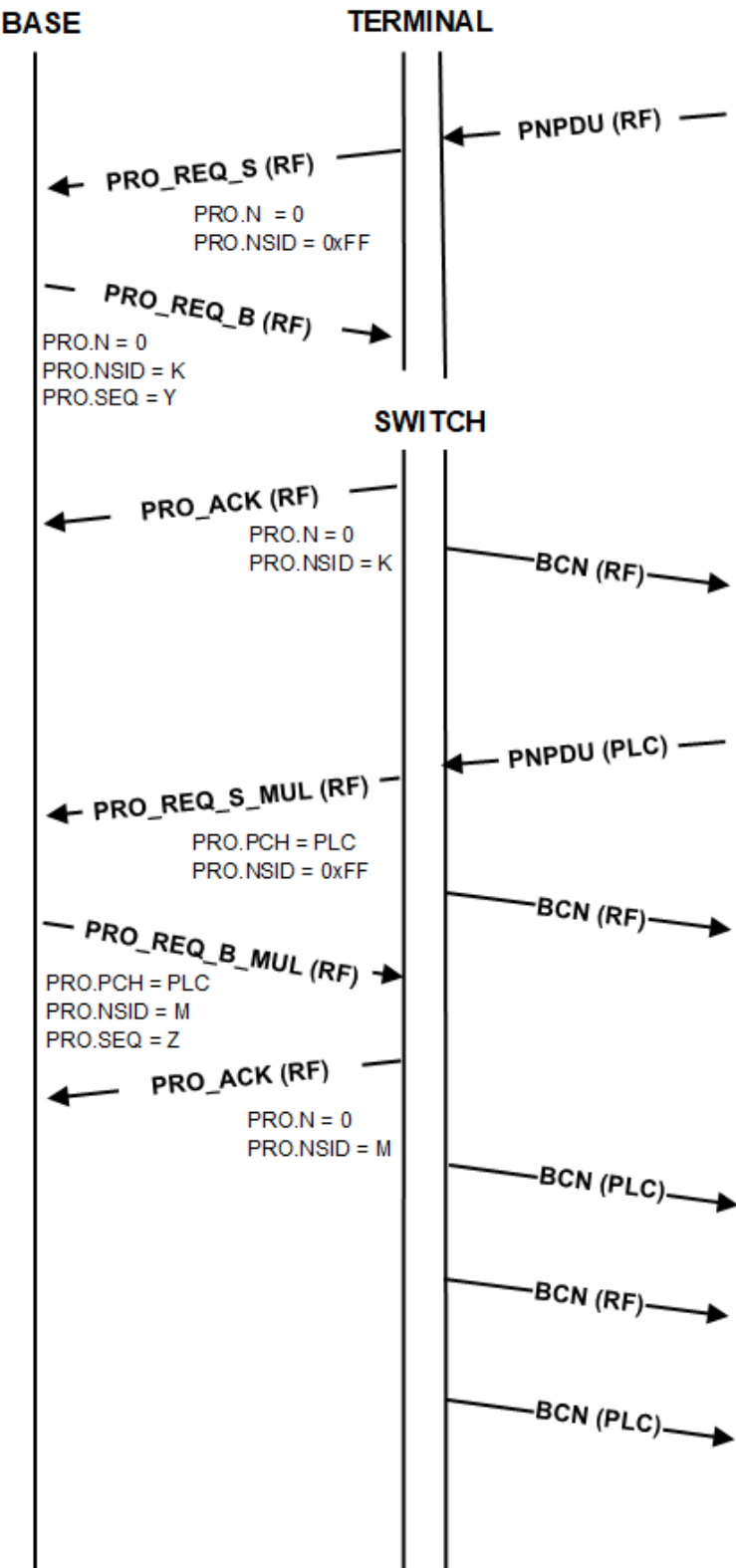


Figure 95 – Two promotion processes initiated by Service Node, one with PRO\_REQ\_S the other with PRO\_REQ\_S\_MultiPHY

4.6.4 Demotion process

The Base Node or a Switch Node may decide to discontinue a switching function at anytime. The demotion process provides for such a mechanism. The PRO control packet is used for all demotion transactions.

The PRO.NSID field shall contain the SID of the Switch Node that is being demoted as part of the demotion transaction. The PRO.PNA field is not used in any demotion process transaction and its contents are not interpreted at either end. PRO.MOD field is not used, it shall be set to zero and not interpreted at either end.

Following successful completion of a demotion process, a Switch Node shall immediately stop the transmission of beacons and change from a *Switch* functional state to a *Terminal* functional state (see Figure 96 and Figure 97). Nodes that support Multi-PHY promotion (see 4.6.3.3) may be Switch on two media. In this case, the Switch Node will immediately stop the transmission of beacons in the medium interested by the successful demotion process (which is uniquely associated to a PRO.NSID) keeping the Switch functional state on the other medium (see Figure 98 and Figure 99).

The present version of this specification does not specify any explicit message to reject a demotion requested by a peer at the other end.

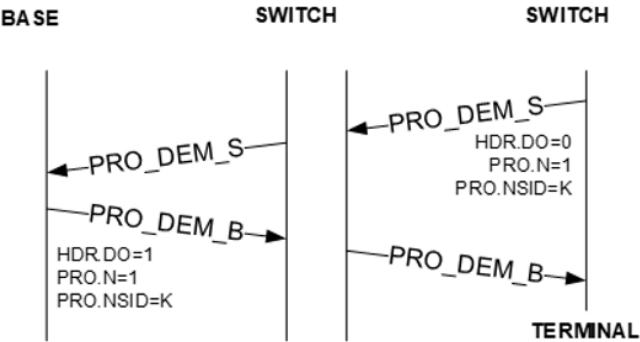


Figure 96 – Demotion process initiated by a Service Node. From Switch to Terminal.

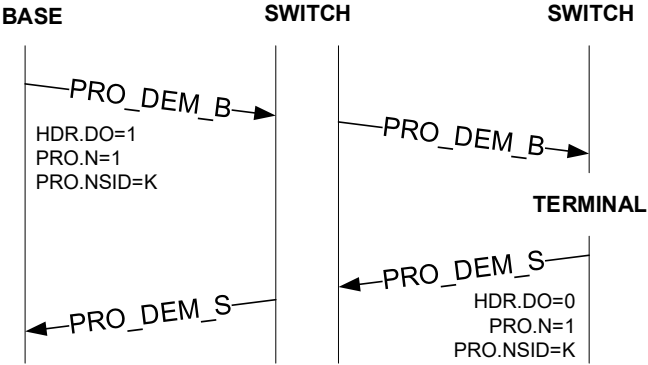


Figure 97 – Demotion process initiated by the Base Node. From Switch to Terminal.

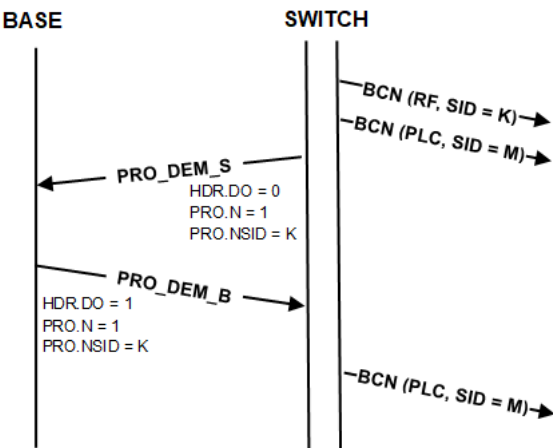


Figure 98 - Demotion process initiated by a Service Node. From Switch in two media to Switch in one medium.

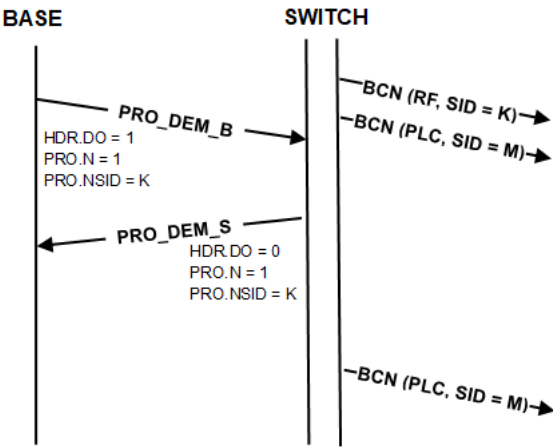


Figure 99 - Demotion process initiated by a Base Node. From Switch in two media to Switch in one medium.

## 4.6.5 Keep-Alive process

### 4.6.5.1 General

The Keep-Alive process is used to perform two operations:

- To detect when a Service Node has left the Subnetwork because of changes to the network configuration or because of fatal errors it cannot recover from.
- To perform robustness management on each hop in the path to the Service Node.

Service Node shall use one timer,  $T_{\text{keep-alive}}$ , to detect if it is no longer part of the Subnetwork. If  $T_{\text{keep-alive}}$  expires, the Service Node assumes it has been unregistered by the Base Node and shall enter in the *Disconnected* functional state. The timer is started when the Service Node receives the REG\_RSP packet with value encoded in the REG.TIME field.

3367 The timer is refreshed when any of the following packets has been received with the TIME information  
3368 provided in those packets:

- 3369 • REG\_RSP packet (repetition).
- 3370 • ALV\_REQ\_B packet.
- 3371 • PRO\_REQ packet.

3372 The timer is also restarted with the last time received in one of the above packets according to the following  
3373 rules:

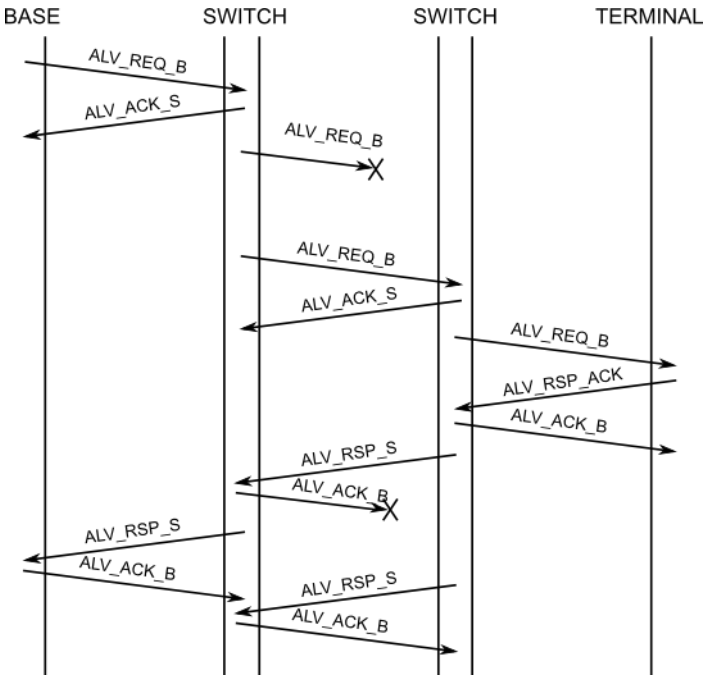
- 3374 • Service nodes which are the final destination of a message, the timer is restarted on reception of
  - 3375 ○ data on an ARQ connection, which fulfills the following conditions: is originated from the
  - 3376 Base Node, is addressed to the node itself and has not yet been acknowledged. Repetitions
  - 3377 of the same packet shall not update the timer.
  - 3378 ○ a CON\_REQ\_B, CON\_CLS\_B, MUL\_JOIN\_B, MUL\_LEAVE\_B or SEC\_REQ control packet which
  - 3379 is addressed to the node itself.
- 3380 • Intermediate Switch nodes restart their timer when transmitting an ALV\_RSP\_S from an ALV
- 3381 procedure of a node below them. The timer is restarted with the last time information received in a
- 3382 REG\_RSP, ALV\_REQ\_B or PRO\_REQ packet addressed to the switch node itself.
- 3383

3384 Each switch along the path to a node keeps track of the switches that are being promoted below it as  
3385 described in section 4.3.4.3

3386 The keep alive process has a link level acknowledge, each switch in the path to the target Service Node is  
3387 responsible for the retransmissions with the next node, up to *macALVHopRepetitions* retransmissions  
3388 (*macALVHopRepetitions* + 1 packets sent). Each retransmission shall be performed in a time equal to a frame  
3389 time. On a reception of an ALV\_REQ\_B/ALV\_RSP\_S the receiving node shall respond with an  
3390 ALV\_ACK\_S/ALV\_RSP\_ACK/ALV\_ACK\_B as soon as possible and with *MACALVACKCtrlPktPriority* priority.  
3391 These retransmissions shall be used to perform robustness management according to section 4.6.8.3.

3392 If the Service Node identifies that the received ALV\_REQ\_B/ALV\_RSP\_S is a retransmit of an already received  
3393 packet, the node shall send the related ACK but shall not switch the Alive to the next hop (since it already did  
3394 it). The algorithm to detect this situation is up to the manufacturer, as a guideline, it could store the last ALV  
3395 operation's data and check if it matches.

3396

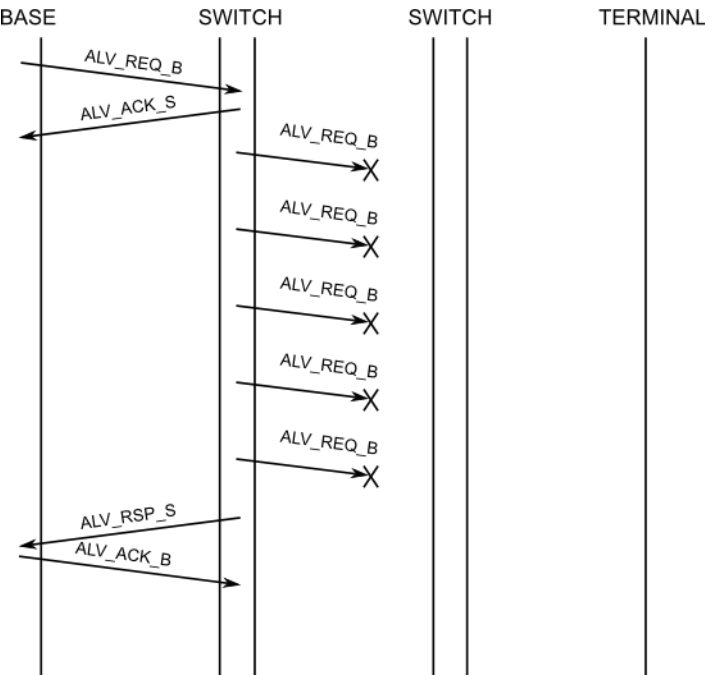


3397

3398

Figure 100 - Successful ALV procedure

3399 If the retransmissions reach the maximum during ALV\_REQ\_B process, the switch node shall start the  
3400 ALV\_RSP\_S procedure.



3401

3402

Figure 101 - Failed ALV procedure

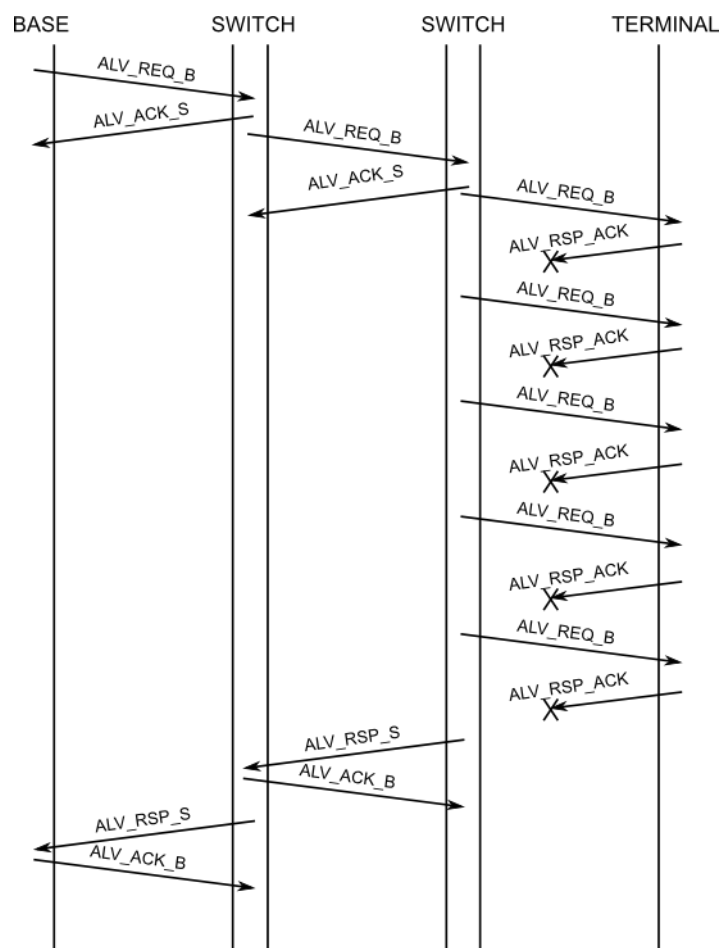


**Figure 102 - Failed ALV procedure (uplink)**

Every time a switch performs a retransmission, it shall decrease the ALV.RTL field before sending the packet, and fill the record of local retransmissions accordingly. When ALV.RTL reaches 0 and the switch has to perform a retransmission, it shall discard the packet.

Every switch shall add the information of the retransmissions needed to reach the node in the ALV.REP\_D and ALV.REP\_U fields, filling the array with the retransmissions needed for each link, both downlink and uplink. The Base Node shall form the ALV message with all the registries and the Service Nodes shall fill it with the values. The rule to know in which record a switch shall add its repetitions is the following.

- Uplink
  - If the switch's (or terminal's) level is equal or lower than ALV.MIN\_LEVEL it shall sum its repetitions to the first record.
  - If the switch's (or terminal's) level is greater than ALV.MIN\_LEVEL it shall use the records sequentially from the beginning. A formula to compute which record shall be used: subtract ALV.MIN\_LEVEL from its own level. E.g.: If ALV.MIN\_LEVEL = 2 and the switch is at level 3 it shall use the record ALV.REP\_U(1).
- Downlink
  - Base node always uses the first record (record number 0: ALV.REP\_D(0)).
  - If the switch's level+1 is equal or lower than ALV.MIN\_LEVEL it shall sum its repetitions to the first record (record number 0: ALV.REP\_D(0)/ALV.REP\_U(0)).



- 3423
- 3424
- 3425
- 3426
- 3427
- If the switch’s level+1 is greater than ALV.MIN\_LEVEL it shall use the records sequentially from the beginning. A formula to compute which record shall be used: subtract ALV.MIN\_LEVEL from its own level+1. E.g.: If ALV.MIN\_LEVEL = 2 and the switch is at level 3 it shall use the record ALV.REP\_D(2).

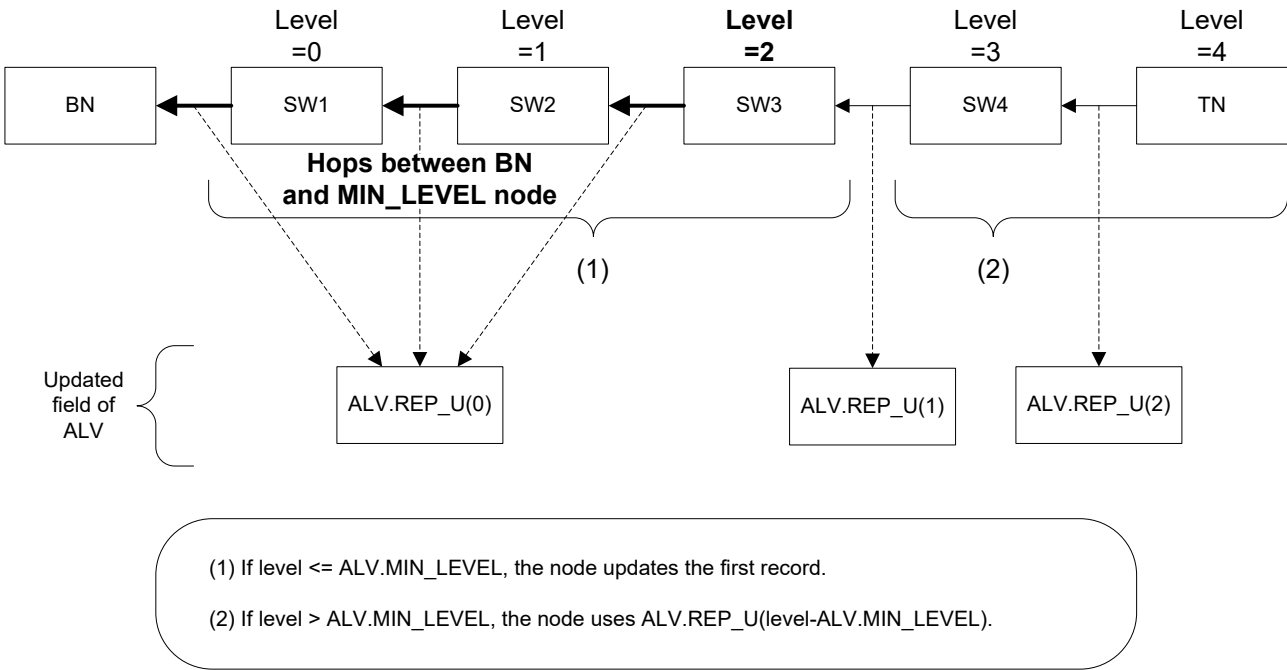


Figure 103 - ALV.REP\_U Example (ALV.MIN\_LEVEL = 2)

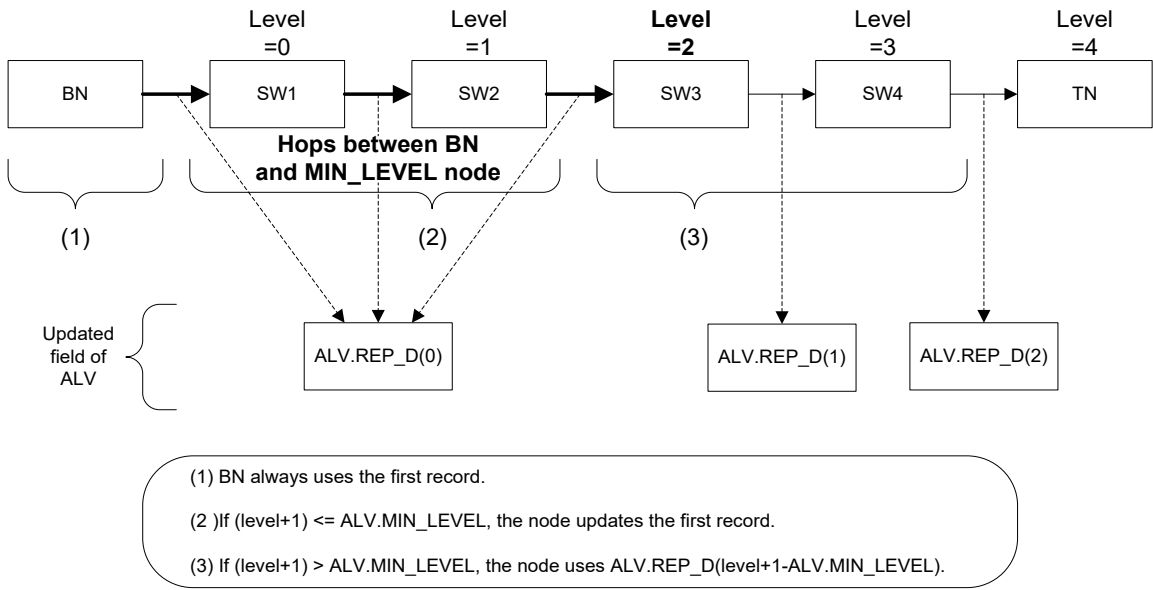


Figure 104 - ALV.REP\_D Example (ALV.MIN\_LEVEL = 2)

The Base Node shall provide the uplink information whenever available in the ALV.REP\_U(\*) fields using the ALV\_REQ\_B message, this provides connection quality information to the service node. If the retransmission at any hop is not available at the time the ALV\_REQ\_B is sent, the Base Node shall set the ALV.VALU(\*) value

3435 to 0 for that hop, and the Service Node shall ignore this record. The first ALV procedure for a hop after  
3436 registration of a Service Node shall be mark as invalid.

3437 At the end of the process the Base Node receives the ALV\_RSP\_S of the last switch with information of the  
3438 connectivity of the node and all the hops in its path. If any of the ALV.VAL\_D or ALV.VAL\_U are marked as  
3439 invalid (equal to 0) or the number of repetitions in ALV.REP\_D or ALV.REP\_U fields are finished in any hop  
3440 (equal to 7), the base node will consider the ALV process as failed.

3441 This operation is more robust than round-trip control packet transaction (CON, REG), so the base node can  
3442 decide that the node does not have enough connectivity and start an unregistration process with it.

3443 The algorithm used by the Base Node to determine when to send ALV\_REQ\_B messages to registered Service  
3444 Nodes and how to determine the value ALV.TIME, PRO.TIME and REG.TIME is left to implementers.

3445 A Switch Node is required to be able to queue *MACConcurrentAliveProcedure* of each ALV\_REQ\_B and  
3446 ALV\_RSP\_S messages at a time. The base node is shall space the ALV\_REQ\_B queries appropriately.

#### 3447 **4.6.5.2 Use of legacy PRIME 1.3.6 keep-alive mechanism**

3448 In some particular network configurations, including exclusively PLC-only nodes, the use of legacy PRIME  
3449 1.3.6 Keep-Alive process instead of the PRIME 1.4 Keep-Alive process can be required. For this reason, all  
3450 PRIME 1.4 PLC-only implementations must be able to implement support for REG.ALV\_F field (Section  
3451 4.4.2.6.3) in REG control packet. This implies that the Base Node shall have the ability to move the entire  
3452 Subnetwork to ALV procedure listed in Section K.2.5. The Subnetwork Keep-Alive process in use can be  
3453 determined by reading the Base Node PIB attribute *macAliveTimeMode* (Section 6.2.3.2). The configuration  
3454 of the Keep-Alive process is left to Base Node implementations.

3455 For simplification, PLC+RF and RF-only nodes support one alive process, i.e. the PRIME 1.4 Keep-Alive. Hence,  
3456 Subnetworks including PLC+RF and/or RF-only nodes are managed using the PRIME 1.4 Keep-Alive process.  
3457 For these subnetworks, the Base Node shall insure that the REG.ALV\_F field in REG control packet is set to  
3458 1. Similarly, the Base Node PIB attribute *macAliveTimeMode* is set to 0.

#### 3459 **4.6.6 Connection establishment**

3460 Connection establishment works end-to-end, connecting the application layers of communicating peers.  
3461 Owing to the tree topology, most connections in a Subnetwork will involve the Base Node at one end and a  
3462 Service Node at the other. However, there may be cases when two Service Nodes within a Subnetwork need  
3463 to establish connections. Such connections are called direct connections and are described in section 4.3.6.

3464 All connection establishment messages use the CON control packet. The various control packets types and  
3465 specific fields that unambiguously identify them are given in Table 24.

3466 Each successful connection established on the Subnetwork is allocated an LCID. The Base Node shall allocate  
3467 an LCID that is unique for a given LNID.

3468 **Note.** *Either of the negotiating ends may decide to reject a connection establishment request. The receipt of*  
3469 *a connection rejection does not amount to any restrictions on making future connection requests; it may*  
3470 *however be advisable.*

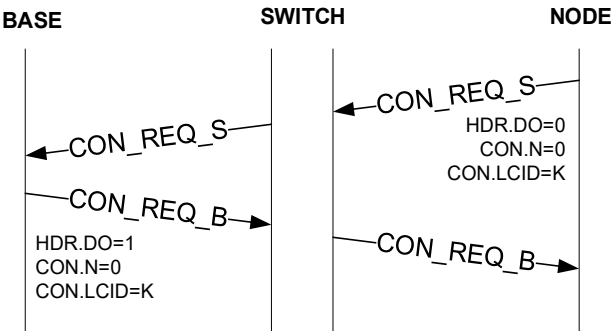


Figure 105 – Connection establishment initiated by a Service Node

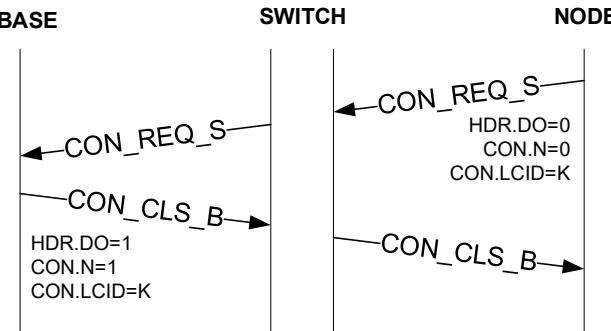


Figure 106 – Connection establishment rejected by the Base Node

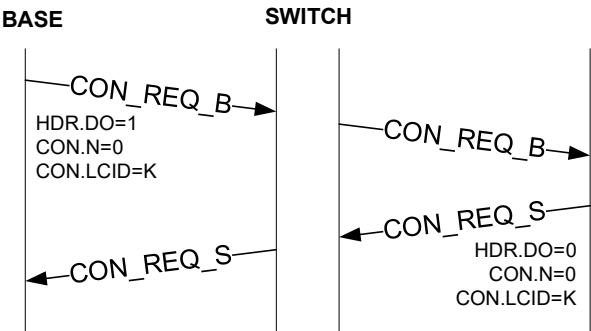


Figure 107 – Connection establishment initiated by the Base Node

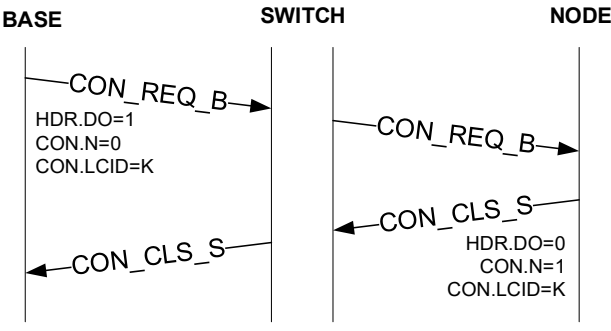


Figure 108 – Connection establishment rejected by a Service Node

#### 4.6.6.1 Connection closing

Either peer at both ends of a connection may decide to close the connection at anytime. The CON control packet is used for all messages exchanged in the process of closing a connection. The relevant CON control

packet fields in closing an active connection are CON.N, CON.LCID and CON.TYPE. All other fields shall be set to 0x0.

A connection closure request from one end is acknowledged by the other end before the connection is considered closed. The present version of this specification does not have any explicit message for rejecting a connection termination requested by a peer at the other end.

Figure 109 and Figure 110 show message exchange sequences in a connection closing process.

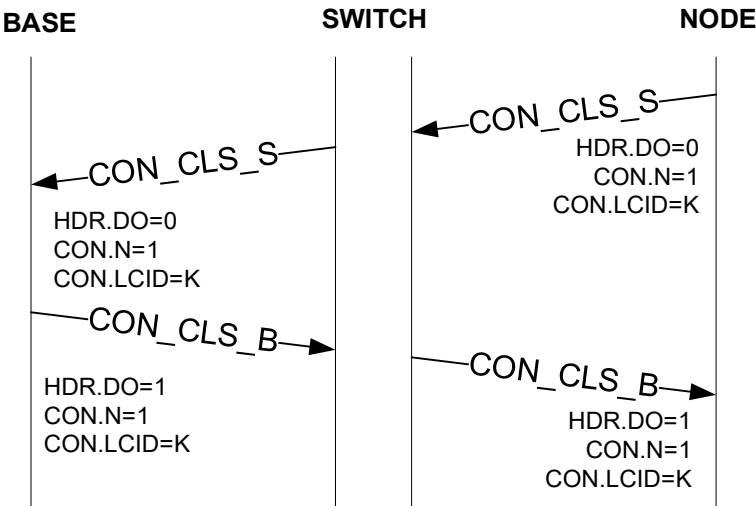


Figure 109 – Disconnection initiated by a Service Node

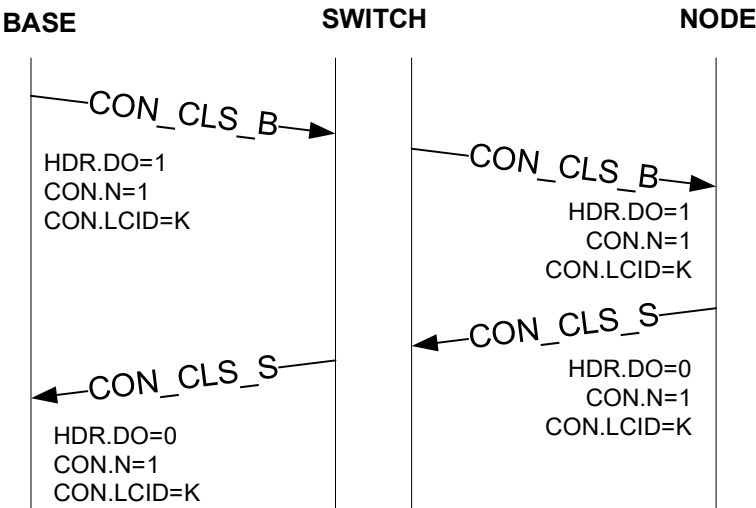


Figure 110 – Disconnection initiated by the Base Node

## 4.6.7 Multicast group management

### 4.6.7.1 General

The joining and leaving of a multicast group can be initiated by the Base Node or the Service Node. The MUL control packet is used for all messages associated with multicast and the usual retransmit mechanism for control packets is used. These control messages are unicast between the Base Node and the Service Node.

**4.6.7.2 Group Join**

Multicast group join maybe initiated from either the Base Node or Service Node. A device shall not start a new join procedure before an existing join procedure started by itself is completed.

Certain applications may require the Base Node to selectively invite certain Service Nodes to join a specific multicast group. In such cases, the Base Node starts a new group and invites Service Nodes as required by application.

Successful and failed group joins initiated from Base Node are shown in Figure 111 and Figure 112.

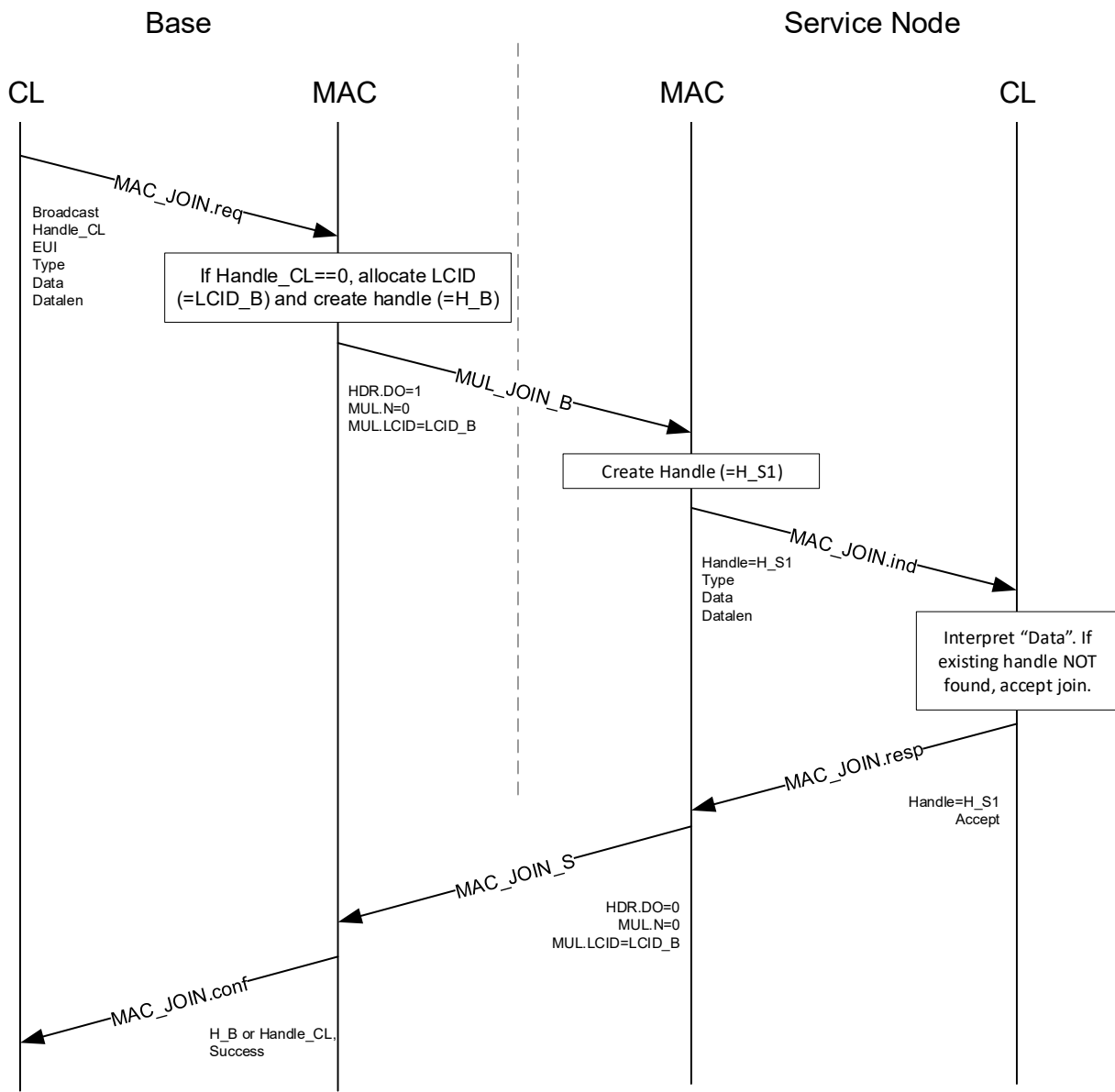


**Figure 111 – Successful group join initiated by Base Node**

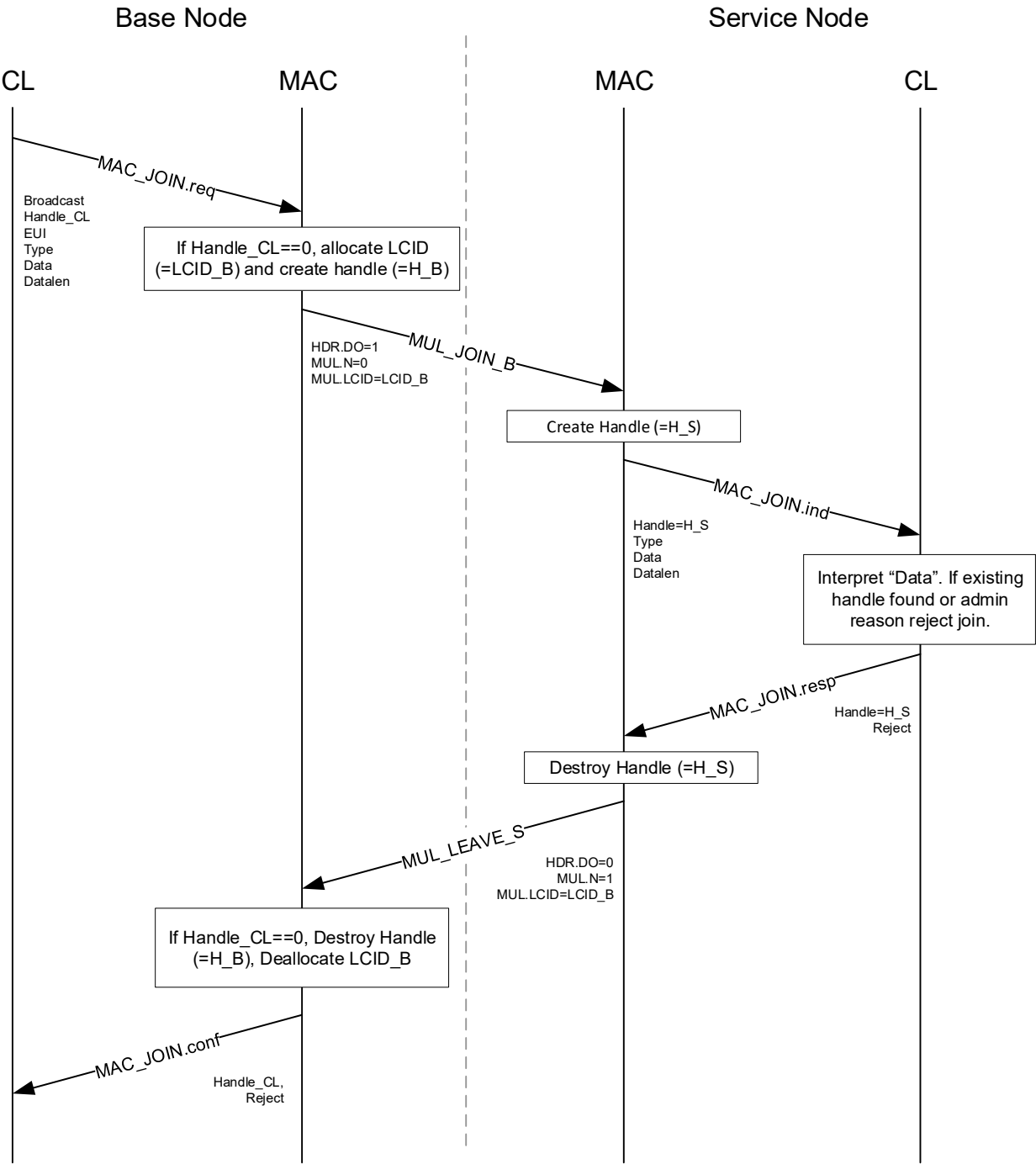
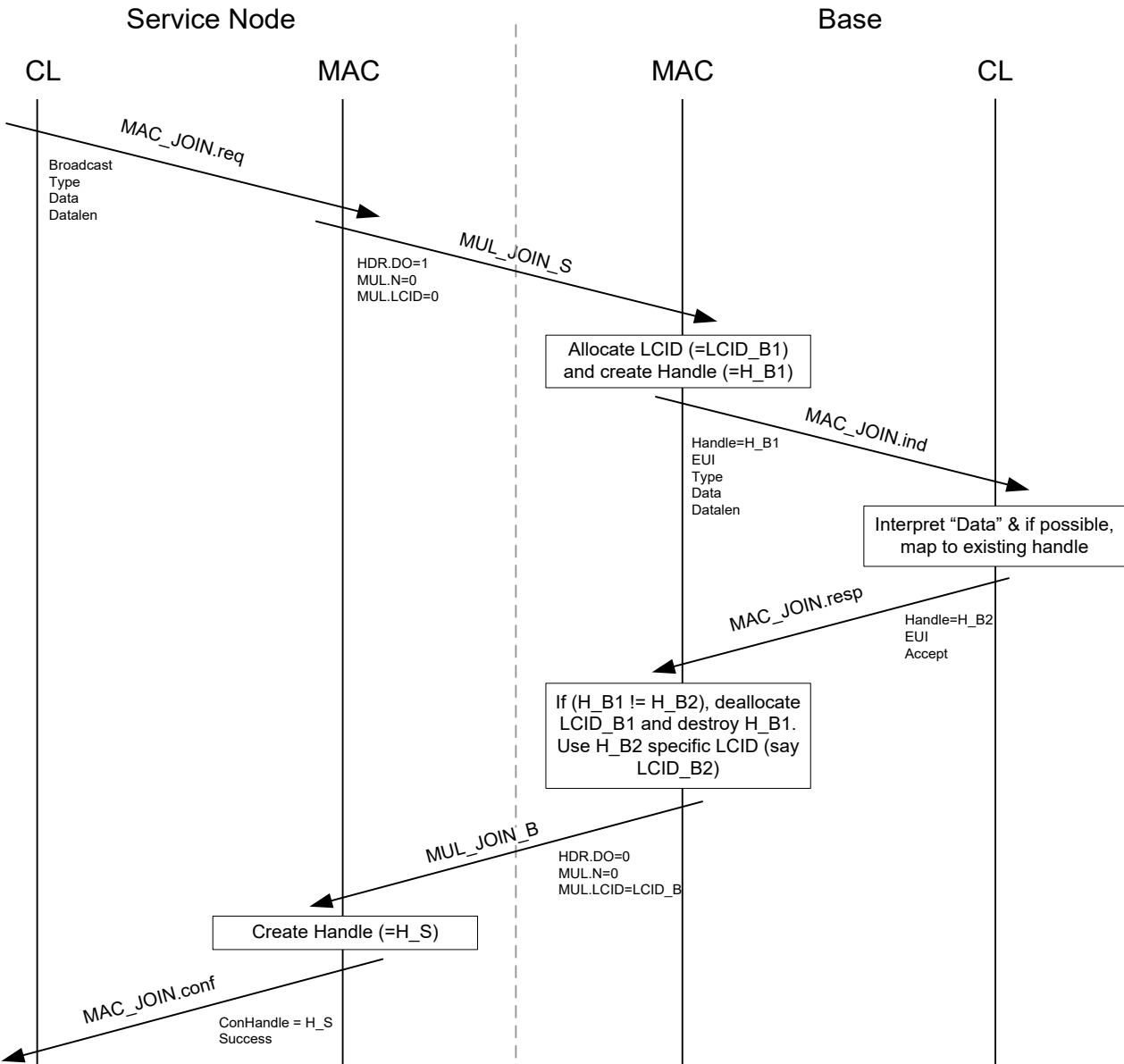


Figure 112 – Failed group join initiated by Base Node

Successful and failed group joins initiated from Service Node are shown in Figure 113 and Figure 114

3510



3511

3512

Figure 113 – Successful group join initiated by Service Node

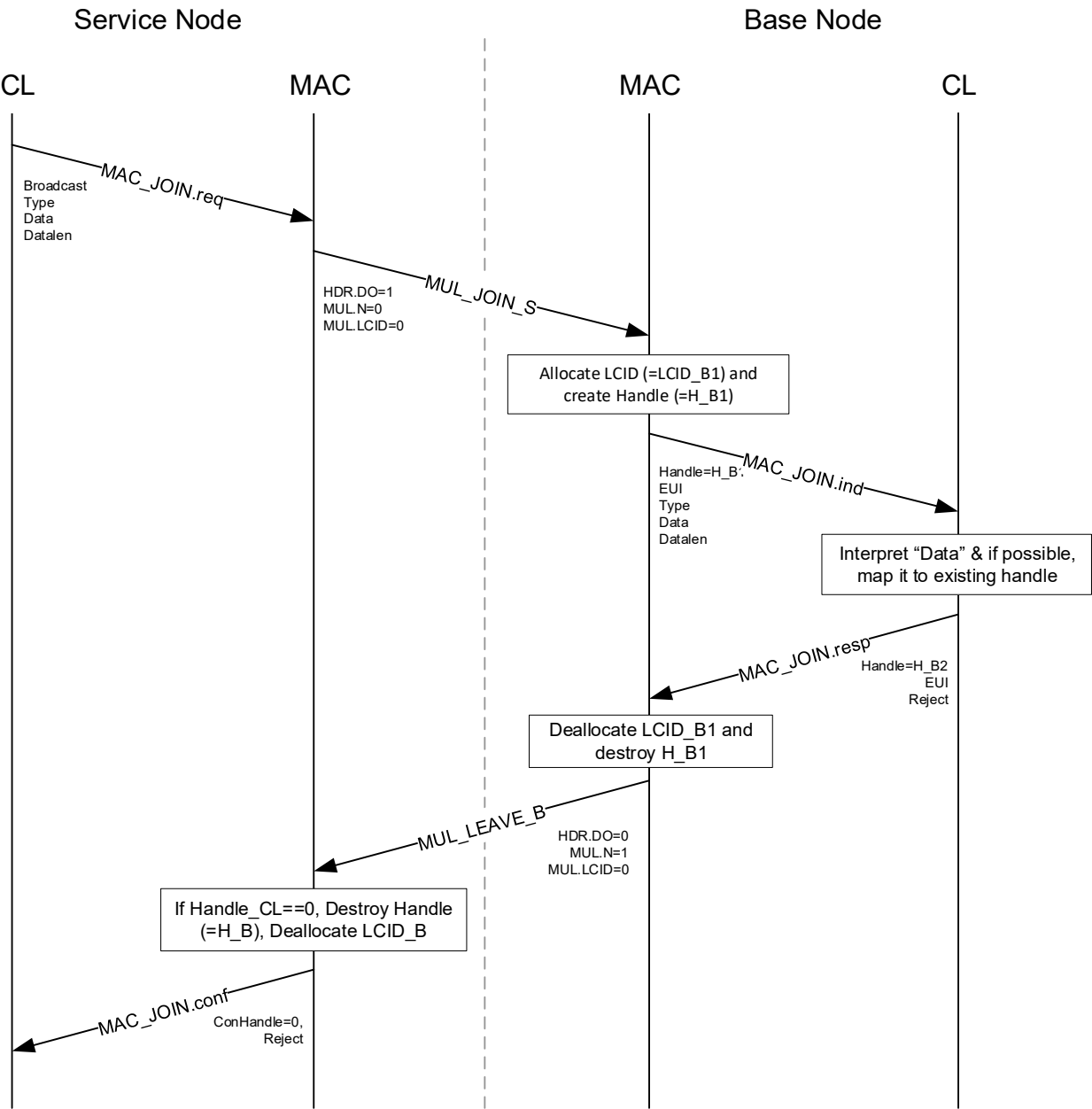


Figure 114 – Failed group join initiated by Service Node.

4.6.7.3 Group Leave

Leaving a multicast group operates in the same way as connection removal. Either the Base Node or Service Node may decide to leave the group. A notable difference in the group leave process as compared to a group join is that there is no message sequence for rejecting a group leave request.



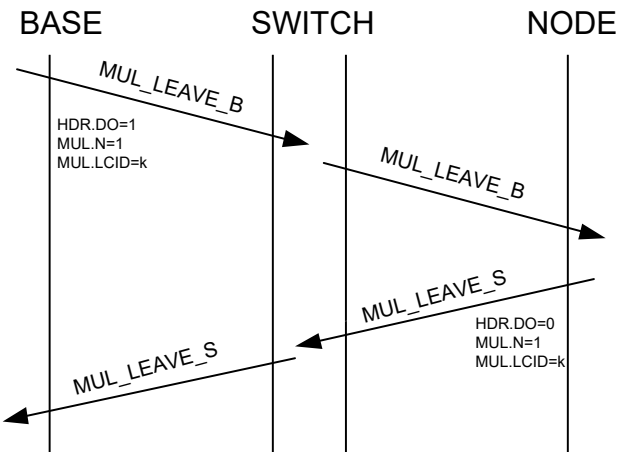


Figure 115 – Leave initiated by the Base Node

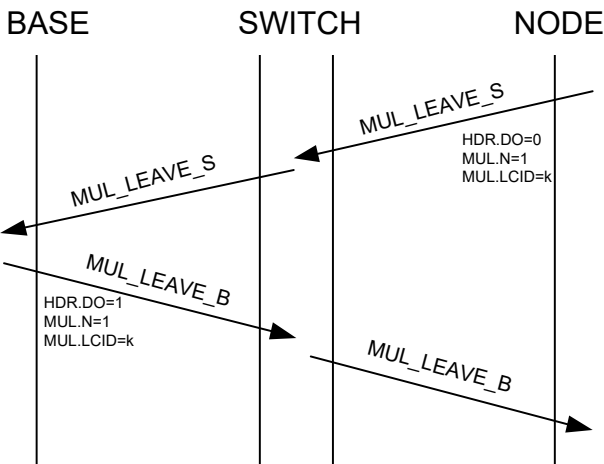


Figure 116 – Leave initiated by the Service Node

4.6.7.4 Multicast Switching Tracking

Switch Nodes need to be aware of the multicast groups under their switching domain. Instead of having to store all the tracking information on the switches themselves, they just need to have a simple multicast table which is managed by both the Switch Node and the Base Node.

Switch Nodes should just monitor muticast join operations through MUL\_JOIN messages in order to start switching multicast traffic, and monitor MUL\_SW\_LEAVE messages in order to stop switching multicast traffic.

4.6.7.4.1 Multicast Switching Tracking for Group Join

The following rules apply for switching of traffic on a multicast group join:

- On a successful group join from a Service Node in its control hierarchy, a Switch Node adds a new multicast Switch entry for the group LCID, where necessary. For this purpose, MUL\_JOIN messages are used.

- From that moment on, the Switch Node will switch multicast traffic for that LCID, and stops keeping track of any control message related to that group.
  - The Base Node shall track all the Switch Nodes that switch multicast traffic for every multicast group.
- Figure 117 exemplifies the process and interactions, for the both cases of the group join initiated by the Base Node and by the Service Node and complements the processes illustrated in the figures of section 4.6.7.2.

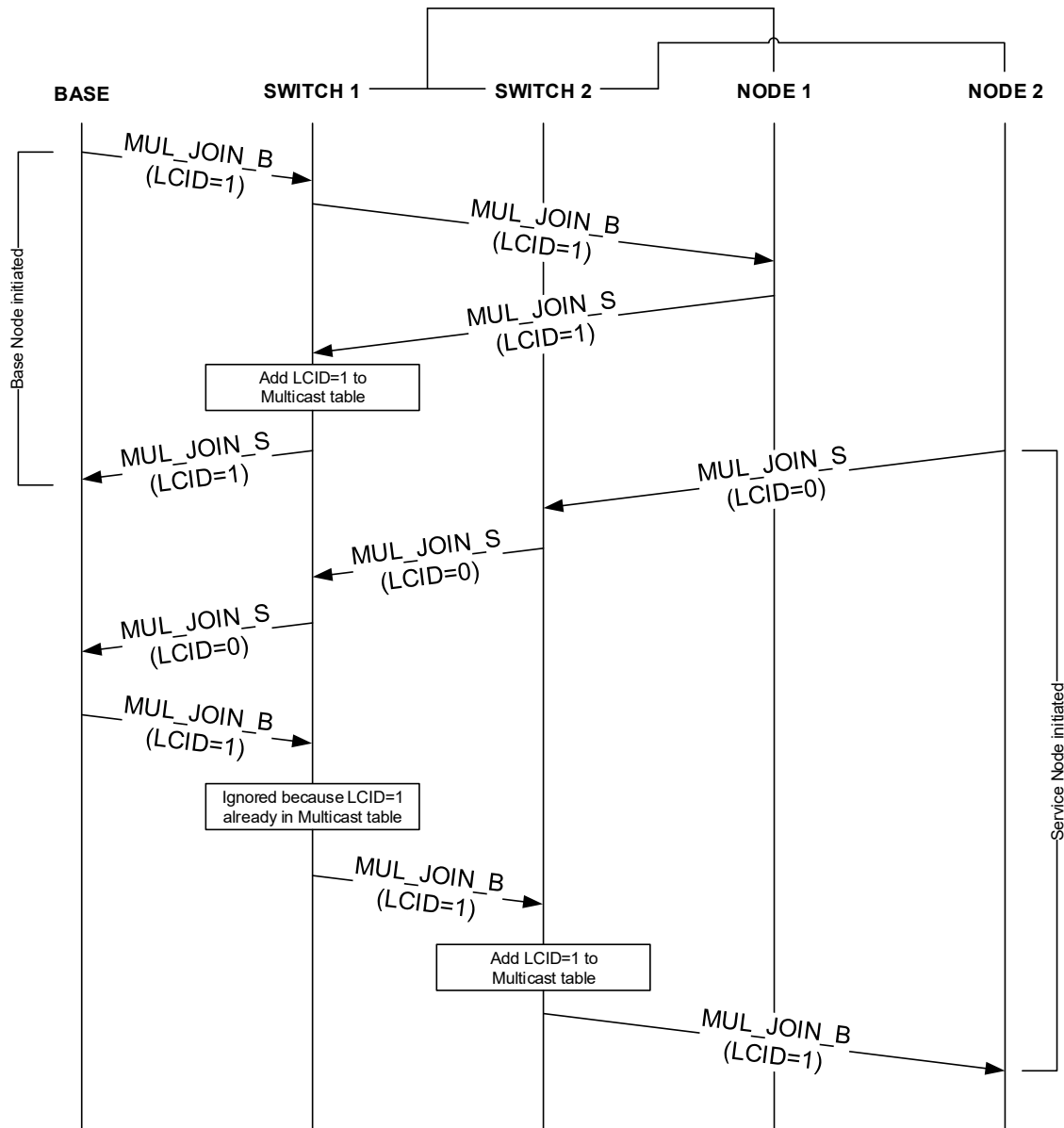


Figure 117 - Multicast Switching Tracking for Group Join.

#### 4.6.7.4.2 Multicast Switching Tracking for Group Leave

For switching of traffic on a multicast group leave, the Base Node shall monitor when all nodes depending on a Switch Node leave a given multicast group, and start a `MUL_SW_LEAVE` procedure to remove that multicast entry for that Switch Node and group.

Figure 118 exemplifies the process and interactions; when they are executed, they take place after the processes illustrated in the figures of section 4.6.7.3.

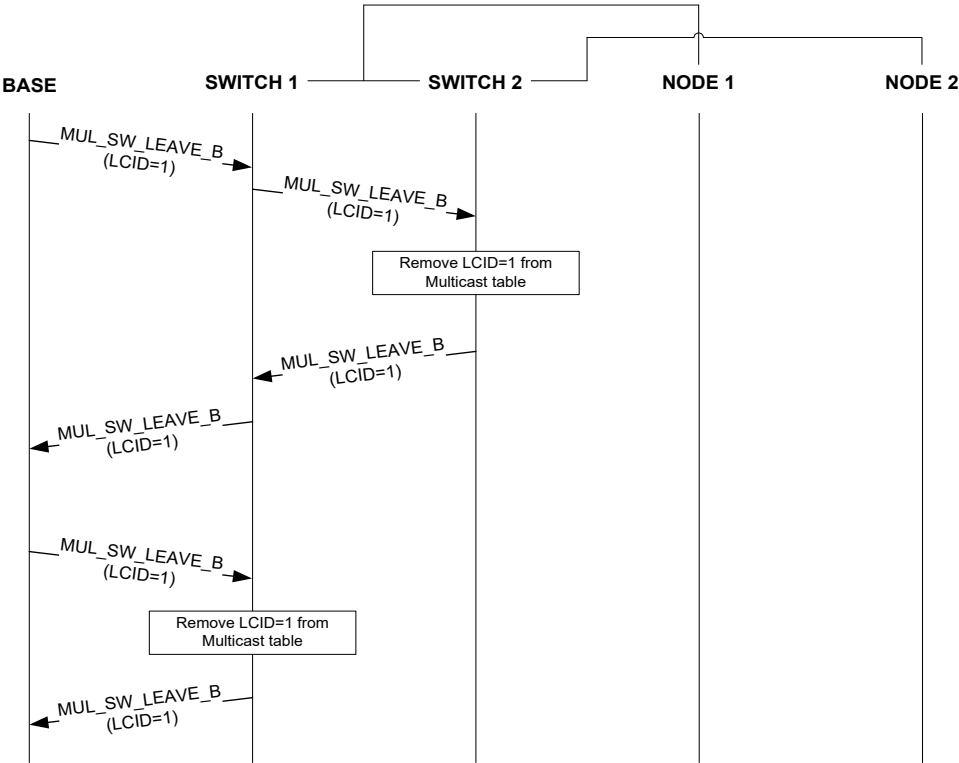


Figure 118 - Multicast Switching Tracking for Group Leave.

4.6.7.4.3 Multicast Switching with double switching

A Switch Node that is transmitting BPDUs on both Type A and Type B PHY frames can switch all data arriving on a multicast connection using both types of PHY frames. This implies replicating data while transmitting twice but this will enable coverage across its entire control domain. Future versions of this specification can further optimize on this to avoid some unwanted traffic that maybe generated by taking this generic approach. This version leaves it open for implementations to either optimize their decision process or replicate data using both modulation schemes.

No matter the policy implemented by the double switch, the device that receives the same broadcast or multicast packet with both modulations shall be able to discard one replica (see section 4.6.3.2).

On receipt of a MUL\_SW\_LEAVE message, the Switch Node shall stop further switching of multicast data for the corresponding connection, on both PHY frame types.

4.6.8 Robustness Management

4.6.8.1 General

The Robustness-management (RM) mechanism is designed on the PLC medium to select the most suitable transmission scheme from the eight available ones (Robust DBPSK, Robust DQPSK, DBPSK\_CC, DBPSK, DQPSK\_CC, DQPSK, D8PSK\_CC and D8PSK). Depending on the transmission channel conditions, the nodes shall decide either to increase the robustness or to select faster transmission modes.

3571 Note that the mechanism described here shall be used to decrease and increase the robustness of Generic  
3572 DATA packets. MAC control packets shall be transmitted in conformance with specification in Section  
3573 4.3.3.3.4.

3574 By default, decision about applicable transmission mode is taken locally. That is, dynamic adaptation of the  
3575 transmission mode is performed taking into account link level channel information, which is exchanged  
3576 between any pair of nodes in direct vision (parent and child). As an exception to this rule, a Base Node may  
3577 decide to disable dynamic robustness-management and force a specific transmission mode in the Service  
3578 Node(s). This static configuration shall be fixed during registration, as explained in 4.6.1.

3579 The robustness-management mechanism comprises two main features:

- 3580 • Link quality information embedded in the packet header of any Generic packets
- 3581 • Link level ACK-ed ALIVE mechanism, as explained in 4.6.8.3 and 4.6.5.

#### 3582 **4.6.8.2 Link quality information embedded in the packet header**

3583 All Generic packets shall convey link quality related information. Four bits in the packet header - “PKT.RM”,  
3584 see 4.4.2.3 – are used by the transmitting device to notify the other peer of the weakest modulation scheme  
3585 that the transmitter considers it could receive. The transmitting device calculates this value processing the  
3586 received packets sent by the other peer. The calculation of PKT.RM value is implementation dependent.

3587 Whenever a node receives a Generic packet from a peer, it shall update the peer related info contained in  
3588 *macListPhyComm* (and *macListMPPHyComm* when supported) PIB as follows:

- 3589 • Store “PKT.RM” from the received packet in *macListPhyComm.phyCommTxModulation*
- 3590 • Reset *macListPhyComm.phyCommRxAge* (time [seconds] since the last update of  
3591 *phyCommTxModulation*).

3592 Whenever a node wants to transmit DATA to an existing peer, it shall check validity of the robustness-  
3593 management information it stores related to that peer. The maximum amount of time that robustness-  
3594 management information is considered to be valid without any further update is specified in PIB  
3595 *macUpdatedRMTimeout*. Consequently, the node shall compare *phyCommRxAge* (time since the last update)  
3596 with *macUpdatedRMTimeout*:

- 3597 • *macListPhyComm.phyCommRxAge*  $\geq$  *macUpdatedRMTimeout*: The node shall transmit using the  
3598 most robust modulation scheme available for the PHY frame type in use. Note: the first time a node  
3599 sends DATA to one peer, RM information is automatically considered to be “out of date” and  
3600 consequently the most robust modulation scheme available shall be used.
- 3601 • *macListPhyComm.phyCommRxAge*  $<$  *macUpdatedRMTimeout*: The node shall check the value  
3602 stored in the *phyCommTxModulation* field:
  - 3603 ○ Different from 0xF: The modulation to be used shall be the same as specified in  
3604 *phyCommTxModulation*.
  - 3605 ○ Equal to 0xF: The node shall transmit using the most robust modulation scheme available  
3606 for the PHY frame type in use.

#### 3607 4.6.8.3 Link level ACK-ed ALIVE mechanism

3608 Alive procedure defines repetitions that are performed in every hop as described in 4.6.5. An  
3609 ALV\_REQ\_B/ALV\_RSP\_S transmitting device shall use this fact to assume a delivery failure if it does not  
3610 receive the corresponding ACK packet. In this case the transmitting device shall re-transmit the packet: the  
3611 first repetition shall be performed with the same robustness, which will be successively increased after  
3612 every link level repetition. Once the maximum number of repetitions is reached, the most robust  
3613 modulation in which the node can transmit shall be stored for that link, even if the repetitions were due to  
3614 the ACK packets.

3615 The device receiving the ALV\_REQ\_B/ALV\_RSP\_S, on reception of a packet being sent more than twice  
3616 (ALV.TX\_SEQ > 1), shall send the ACK packet with at least the same robustness as the received packet.

3617 The ALV packets shall be transmitted in one of the following encodings: DBPSK\_CC, Robust DQPSK and  
3618 Robust DBPSK. The robustness increase should be performed in that order.

3619 In the Terminal Node a three way handshake is performed, once the ALV\_REQ\_B has arrived the Service  
3620 Node shall start a regular ALV\_RSP\_S send transaction following the same rules.

3621 In every case the ALV.RX\_SNR, ALV.RX\_POW and ALV.RX\_ENC shall send those PHY parameters of the last  
3622 received ALV\_REQ\_B/ALV\_RSP\_S packet, and in the PKT.RM they shall send the least robust modulation in  
3623 which it should be able to receive.

#### 3624 4.6.8.4 PHY robustness changing

3625 From the PHY point of view there are several parameters that may be adjusted and which affect the  
3626 transmission robustness: the transmission power and modulation parameters (convolutional encoding and  
3627 constellation). As a general rule the following rules should be followed:

- 3628 • **Increase robustness:** increase the power and, if it is not possible, improve the modulation scheme  
3629 robustness (reducing throughput).
- 3630 • **Reduce robustness:** reduce the modulation scheme robustness (increasing throughput) and, if it is  
3631 not possible, reduce the transmission power.

#### 3632 4.6.9 Channel allocation

3633 Allocation of specific channel resources is possible in the CFP. Each MAC frame shall include a contention free  
3634 period with a minimum duration of (*MACBeaconLength1* + 2 x *macGuardTime*), which may be used for  
3635 beacon transmission and/or allocation of specific application data transmissions. Any kind of CFP usage,  
3636 either beacon transmission or allocation of channel resources for data, shall be always granted by the Base  
3637 Node.

##### 3638 4.6.9.1 Beacon channel allocation

3639 As part of a promotion procedure, a Terminal node may be promoted to Switch status and gain the ability to  
3640 transmit its own beacons. These beacons shall be allocated in the CFP as explained in 4.6.3.

**4.6.9.2 Data channel allocation**

A CFP allocation / de-allocation request to transport application data may be initiated either by the Base Node or the Service Node. The CFP MAC control packet described in 4.4.2.6.7 shall be used for that purpose. Figure 119 below shows a successful channel allocation sequence. All channel allocation requests initiated by Service Node are forwarded to the Base Node. Note that in order to assure a contention-free channel allocation along the entire path, the Base Node allocates non-overlapping times to intermediate Switch Nodes. In a multi-level Subnetwork, the Base Node may also reuse the allocated time at different levels. While reusing the said time, the Base Node needs to ensure that the levels that use the same time slots have sufficient separation so that there is no possible interference.

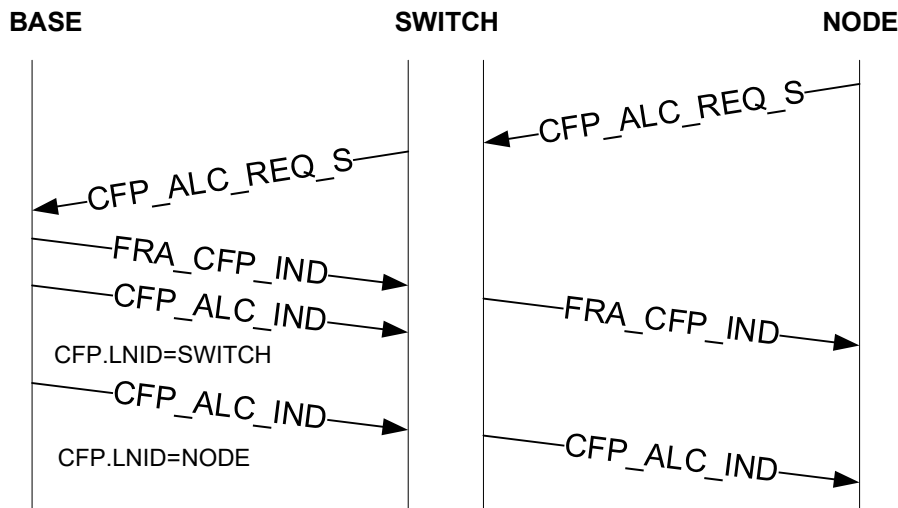


Figure 119 – Successful allocation of CFP period

Figure 120 below shows a channel de-allocation request from a Terminal device and the resulting confirmation from the Base Node.

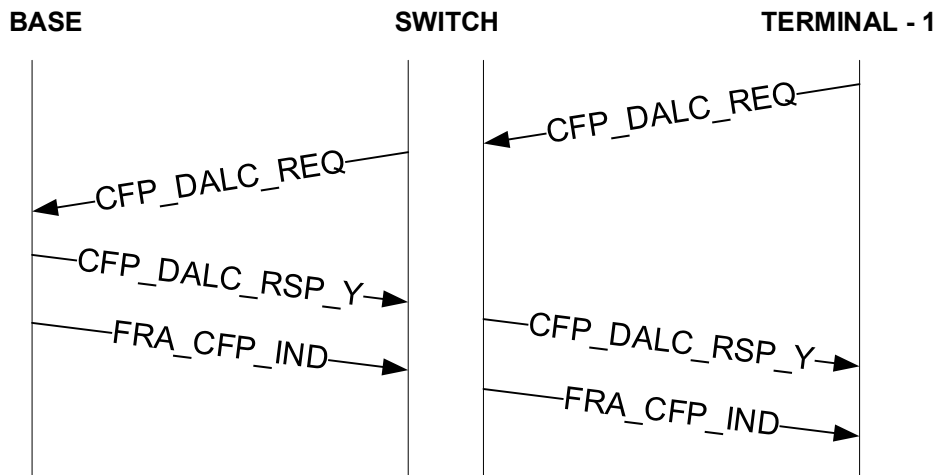
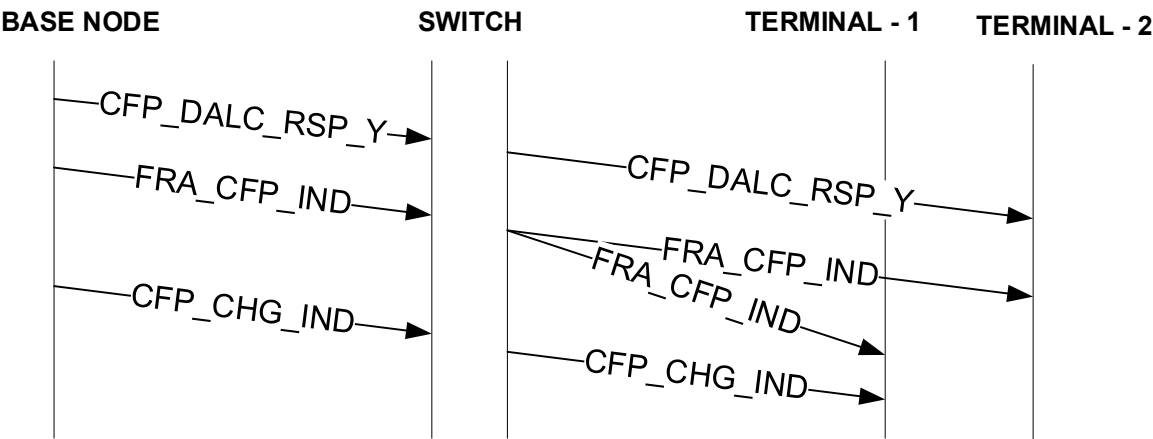


Figure 120 – Successful channel de-allocation sequence

Figure 121 below shows a sequence of events that may lead to a Base Node re-allocation contention-free slot to a Terminal device that already has slots allocated to it. In this example, a de-allocation request from Terminal-2 resulted in two changes: firstly, in global frame structure, this change is conveyed to the

3659 Subnetwork in the FRA\_CFP\_IND (a standard FRA packet intended to change CFP duration only) packet;  
3660 secondly, it is specific to the time slot allocated to Terminal-1 within the CFP.



3661  
3662 **Figure 121 – Deallocation of channel to one device results in the change of CFP allocated to another**

3663 **4.6.10 Programmed Configuration Change**

3664 In addition to indication events associated to FRA\_CFP\_IND/FRA\_BCN\_IND control packets, the Base Node  
3665 can inform the network about a future change of other relevant parameters at network level using the PCC  
3666 (Programmed Configuration Change) MAC control packet.

3667 An example is a change in the Physical layer channel/band used by Base Node on a specific medium to  
3668 communicate with directly connected child Nodes. The PCC MAC control packet described in 4.4.2.6.11 shall  
3669 be used to inform all the network about the change. This information can be used by Service Nodes to  
3670 automatically change their own Physical Layer channel/band in order to minimize the impact of the change  
3671 at MAC level. The actions to be performed in Service Nodes following the reception of a PCC\_PHY\_CH packet  
3672 are specific to the implementation.

3673 **4.6.11 Channel hopping**

3674 Channel hopping is a feature configured at network level for the RF PHY medium, i.e. in a Prime network all  
3675 the nodes use it or not. The channel hopping mechanism uses network coordination with the MAC frame, via  
3676 a shared hop sequence to which devices synchronize. This synchronization is performed upon the reception  
3677 of a BCN frame, which contains all the information necessary for a node to generate the pseudo-random hop  
3678 sequences shared by all nodes within the network, thus allowing the node to communicate with other nodes  
3679 within the network.

3680 Two network pre-shared channel lists, macHoppingInitialChannelList and macHoppingBCNInitialChannelList,  
3681 are used for channel hopping. Depending on choices performed at network level, these lists may be disjoint  
3682 or may have common elements.

3683 From these two lists, two channel hopping sequences are generated:

- 3684
- The main channel sequence is used for all the packets transmitted in the SCP (see 4.6.11.1 and 4.6.11.2) . It is derived from macHoppingInitialChannelList
  - The beacon channel sequence is used for all the packets transmitted in the CFP (see 4.6.11.3 and 4.6.11.4) . It is derived from macHoppingBCNInitialChannelList
- 3685  
3686  
3687

3688 For the main channel sequence, the dwell time is set to 138 symbols or 309.12 milliseconds, which allows  
3689 for the division of the MAC frame in 2, 4, 6 or 8 slots depending on the configured frame length.

3690 For the beacon channel sequence, the dwell time is equal to the macFrameLength (in symbols) in use in the  
3691 network.

3692 Guard-times of macGuardTime duration are present at the CFP boundary as described in 4.3.3.1.

3693 In addition, to help transitions from one RF channel to another in the main channel sequence, specific RF  
3694 channel hopping guard-times of macGuardTime are also defined at symbols  $138n$  where  
3695  $n=1,...,macFrameLength(in\ symbols)/138 - 1$ .

3696 During guard-times, transmission is forbidden.

3697 Examples of channel transitions with different frame and CFP/SCP lengths can be seen in Annex M.

#### 3698 **4.6.11.1 Main channel sequence generation**

3699 The main network hopping sequence is used in the SCP and is a pseudo-randomly shuffled set of all of the  
3700 channels reported as used for hopping in macHoppingInitialChannelList. The selection of channels used for  
3701 main sequence hopping is implementation-specific and should be pre-shared by all nodes within the network.

3702 The mechanism to generate the main sequence is based on the one defined on clause 6.2.10 of IEEE 802.15.4-  
3703 2015 [28]:

- 3704 • SHUFFLE is a macHoppingSequenceLength-sized array. The contents of this array are equivalent to  
3705 the first macHoppingSequenceLength outputs of a 9-bit linear feedback shift register (LFSR) with  
3706 polynomial  $x^9 + x^5 + 1$  and a starting seed of 255:
  - 3707 ○ In order for different PRIME networks to use different sequences the SNA of the network is used  
3708 as an input for the LFSR.
  - 3709 ○ For each LFSR value generated a byte of the SNA prepended with a 0 is used as input.
  - 3710 ○ The SNA bytes are used in order from least significant to most significant.
  - 3711 ○ If the number of channels is higher than the number of SNA bytes these will be reused as inputs  
3712 cyclically.
  - 3713 ○ Each LFSR output is modulo macHoppingSequenceLength, so that each entry of SHUFFLE is  
3714 between 0 and (macHoppingSequenceLength – 1), inclusive.
- 3715 • CHANNELS is a macHoppingSequenceLength-sized array that is initially populated with the  
3716 monotonically increasing set of channels reported as used for hopping in  
3717 macHoppingInitialChannelList.
- 3718 • CHANNELS is shuffled as per Figure 110. Elements may wind up being swapped multiple times in this  
3719 process.



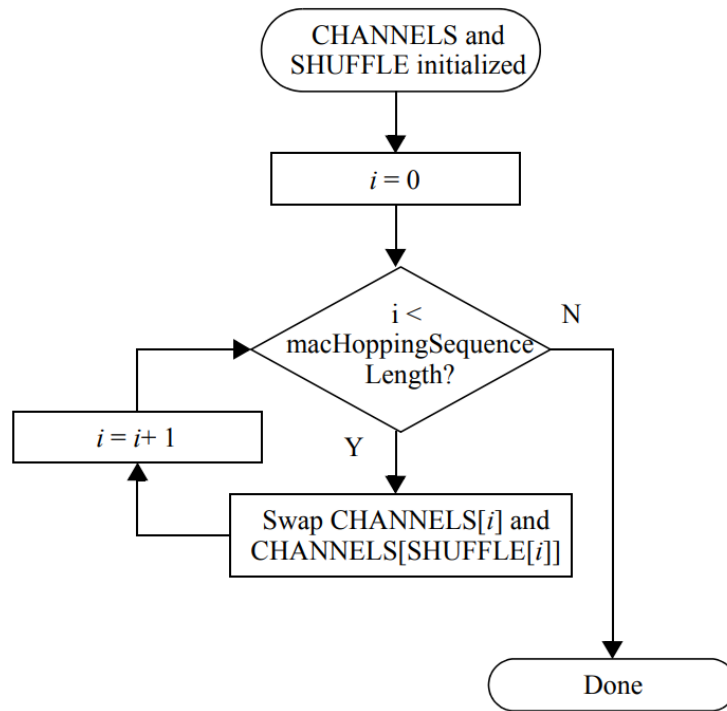


Figure 122 – CHANNELS shuffle algorithm

For cases where `macHoppingSequenceLength` is lower than the number of channels available to the PHY, some channels available to the RF PHY medium will be excluded from the array.

Channel sequence generation examples can be seen in ANNEX M.

#### 4.6.11.2 Main channel sequence and level

For the Base Node the main channel sequence is determined using the following formula:

$$\text{CHANNELS}[\text{mod}(\text{macHoppingSequencePosition}, \text{macHoppingSequenceLength})].$$

In order to optimize the usage of the of the multiple channels of a RF PHY medium, and also to reduce the collision domain, the level of the Service Nodes is used as an offset in the channels sequence. The channel to use at each level will be:

$\text{CHANNELS}[\text{mod}(\text{macHoppingSequencePosition} + \text{level} + 1, \text{macHoppingSequenceLength})]$ . Where `macHoppingSequencePosition` is the current position in the channel sequence. This way two adjacent levels will not use the same frequency channels at any point in time.

This implies that nodes will use different channels for reception and transmission at all times. The nodes will be receiving in their corresponding channel and will have to switch to the previous channel in the sequence for uplink transmission, and to the next channel for downlink transmission.

This also implies that nodes will be unable to receive once they switch channel for transmission.

#### 3738 4.6.11.3 Beacon channel sequence generation

3739 In order to reduce service node start-up times, and to optimize channel usage, a separate channel sequence  
 3740 will be defined for all the frames transmitted in the CFP, including the beacon frames as the most important  
 3741 case. This channel sequence will be generated in the same way as the main channel sequence, but using only  
 3742 a subset (maximum 32 elements) of the total number of channels available to the PHY layer. The subset (and  
 3743 its cardinality) needs to be chosen considering a compromise among simplicity of management, connection  
 3744 time for disconnected nodes, channel usage, specific country regulations. This subset is implementation-  
 3745 specific and should be pre-shared by all nodes within the network. The subset consists of all the channels  
 3746 reported as used for hopping in macHoppingBCNInitialChannelList.

3747 The beacon channel sequence generation mechanism works as follows:

3748 SHUFFLE is a macHoppingBCNSequenceLength-sized array. The contents of this array are equivalent to the  
 3749 first macHoppingBCNSequenceLength outputs of a 9-bit linear feedback shift register (LFSR) with polynomial  
 3750  $x^9 + x^5 + 1$  and a starting seed of 255:

- 3751       ○ In order for different PRIME networks to use different sequences the SNA of the network is used  
 3752       as an input for the LFSR.
- 3753       ○ For each LFSR value generated a byte of the SNA prepended with a 0 is used as input.
- 3754       ○ The SNA bytes are used in order from least significant to most significant.
- 3755       ○ If the number of channels is higher than the number of SNA bytes these will be reused as inputs  
 3756       cyclically.
- 3757       ○ Each LFSR output is modulo macHoppingBCNSequenceLength, so that each entry of SHUFFLE is  
 3758       between 0 and (macHoppingBCNSequenceLength – 1), inclusive.
- 3759       • CHANNELS is a macHoppingBCNSequenceLength-sized array that is initially populated with the  
 3760       monotonically increasing set of channels available in the beacon channel subset.
- 3761       • CHANNELS is shuffled as per Figure 110. Elements may wind up being swapped multiple times in this  
 3762       process.

3763 For the beacon channel sequence macHoppingBCNSequenceLength shall match the number of channels  
 3764 available in the beacon channel subset. This implies the all channels in the subset will appear only once in  
 3765 the beacon channel sequence array.

#### 3766 4.6.11.4 Beacon channel sequence and BCN.SEQ

3767 For all the nodes, the beacon channel sequence is determined using the following formula:

3768 
$$\text{CHANNELS}[\text{mod}(\text{BCN.SEQ}, \text{macHoppingBCNSequenceLength})]$$

3769 As BCN.SEQ is present in every beacon, all the nodes will use the same channels for reception and  
 3770 transmission for beacons (and all CFP packets) at all times. Note that for the Base Node, the formula only  
 3771 refers to the transmitted beacons.

#### 3772 4.6.11.5 Service node start-up and synchronization

3773 Upon starting up, a service node will start scanning for beacons in the different RF channels available in the  
 3774 beacon channel list. The maximum time for a network to send beacons in all channels available is  
 3775 macHoppingBCNSequenceLength times the maximum superframe length (32\*1104 symbols, 79.13472

seconds). It is recommended that service nodes scan each channel for the maximum time required for the network to send beacons in all channels. The algorithm used to decide which channel is being scanned by a service node is left to the implementer.

Upon reception of a BCN frame a service node will generate both the main sequence and the beacon sequence by using the BCN.SNA field. The macHoppingBCNSequencePosition can be derived by the node by reading the BCN.SEQ of the received beacon (see 4.6.11.3). This allows all the nodes to be coordinated on the RF channel used to transmit their RF beacon. For beacons received on the RF medium, the node may also derive the macHoppingBCNSequencePosition looking for the channel where the beacon was received in the beacon sequence.

For the main channel sequence, the macHoppingSequencePosition is included in the BCN.HOP\_POS field, when the beacon is received on the RF medium. The main channel sequence is autonomously started from macHoppingSequencePosition = 0, by a node registered through PLC medium. For the transmission of PNPDU's disconnected nodes will sweep through all available channels in the beacon channel list and the main channel list sending a PNPDU frame in each of them in the shortest possible amount of time.

In order to randomize the channel usage, the channel sequence used for the PNPDU transmission shall be a pseudo-random sequence generated using the previously described algorithm but replacing the SNA with the EUI48 of the node.

Disconnected nodes shall not transmit less than one PNPDU per channel per macHoppingPromotionMaxTxPeriod units of time, and no more than one PNPDU per channel per macHoppingPromotionMinTxPeriod. The algorithm used to decide the transmission rate of PNPDU's is left to the implementer.

## 4.6.12 Network Attenuation Configuration

This section outlines the mechanisms for controlling and applying network-wide attenuation settings in a PLC network. The feature ensures that all nodes within the network adhere to a specified attenuation value relative to their nominal transmission power, thereby harmonizing transmission power levels throughout the network.

### 4.6.12.1 Attenuation Encoding and Transmission

The network attenuation value is encoded across a combination of bits from the Beacon (BCN) and Registration Response (REG\_RSP) messages. The attenuation is defined as a 5-bit value, facilitating a range from 0 to 31 dB of attenuation.

- **BCN.PLC\_ATT:** The two most significant bits of the network attenuation are transmitted within the BCN message. This 2-bit field dictates the initial range of attenuation levels that a node should use during the registration process in steps of 8 dB:
  - **00b:** 0-7 dB.
  - **01b:** 8-15 dB.
  - **10b:** 16-23 dB.
  - **11b:** 24-31 dB.
- **REG.PLC\_ATT:** The REG\_RSP message includes the three least significant bits of the attenuation value (0-7 dB). This completes the 5-bit attenuation value required for final adjustment post-registration.

#### 3815 4.6.12.2 Operational Procedure

3816 Upon reception of a BCN packet, service nodes extract the BCN.ATT value to ascertain the most significant  
3817 part of the attenuation level. Nodes initially attempting to register to a network shall use the attenuation  
3818 value received in the beacon. The node may chose a value of attenuation within the range provided in the  
3819 beacon to send the REG\_REQ message.

3820 The REG\_RSP message provides the remaining bits needed to finalize the attenuation level. The node shall  
3821 use the final attenuation value as long as it remains registered within the network.

3822 The application of attenuation values to the REG.REQ message is deemed implementation-specific. Possible  
3823 strategies may include applying only the initial BCN.PLC\_ATT value for nodes undergoing initial registration  
3824 or incorporating the full attenuation value for previously registered nodes attempting re-registration.

3825 Switch nodes shall propagate the network attenuation by setting the BCN.PLC\_ATT field value to the value  
3826 received from their switch nodes.

3827 The PLC attenuation value used by a node can be obtained reading the macPhyPLCNetworkAttenuation PIB  
3828 attribute.

#### 3829 4.6.12.3 Dynamic power configuration

3830 In order to change the current PLC attenuation of a network a Programmable Configuration Change (PCC)  
3831 process can be used. This process will be triggered in the base node by writing the  
3832 macPhyPLCNetworkAttenuationChange PIB attribute.

3833 The PLC network attenuation PCC message (PCC.TYP=1) includes the complete attenuation value as well as  
3834 the frame sequence when the change shall be applied. Upon its reception, the nodes in the network shall  
3835 schedule the change for the given frame sequence.

3836 In order to reduce the issue of service nodes missing the PLC network attenuation change PCC message,  
3837 service nodes shall unregister from the network if they receive a beacon with a BCN.PLC\_ATT field with a  
3838 different value from the one currently in use.

3839

### 3840 4.7 Automatic Repeat Request (ARQ)

#### 3841 4.7.1 General

3842 Devices complying with this specification may either implement an ARQ scheme as described in this section  
3843 or no ARQ at all. This specification provides for low-cost Switch and Terminal devices that choose not to  
3844 implement any ARQ mechanism at all.

#### 3845 4.7.2 Initial negotiation

3846 ARQ is a connection property. During the initial connection negotiation, the originating device indicates its  
3847 preference for ARQ or non-ARQ in CON.ARQ field. The responding device at the other end can indicate its  
3848 acceptance or rejection of the ARQ in its response. If both devices agree to use ARQ for the connection, all

traffic in the connection will use ARQ for acknowledgements, as described in Section 4.7.3. If the responding device rejects the ARQ in its response, the data flowing through this connection will not use ARQ.

### 4.7.3 ARQ mechanism

#### 4.7.3.1 General

The ARQ mechanism works between directly connected peers (original source and final destination), as long as both of them support ARQ implementation. This implies that even for a connection between the Base Node and a Terminal (connected via one or more intermediate Switch devices), ARQ works on an end-to-end basis. The behavior of Switch Nodes in an ARQ-enabled connection is described in Section 4.7.4. When using ARQ, a unique packet identifier is associated with each packet, to aid in acknowledgement. The packet identifier is 6 bits long and can therefore denote 64 distinct packets. ARQ windowing is supported, with a maximum window size of 32 (5 bits), as described in Section 4.7.3.3.

#### 4.7.3.2 ARQ PDU

##### 4.7.3.2.1 General

The ARQ subheader contains a set of bytes, each byte containing different subfields. The most significant bit of each byte, the M bit, indicates if there are more bytes in the ARQ subheader.

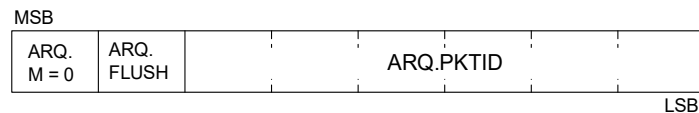


Figure 123 - ARQ subheader only with the packet id

Figure 123 shows an ARQ subheader with the first M bit of 0 and so the subheader is a single byte and contains only the packet ID for the transmitted packet.

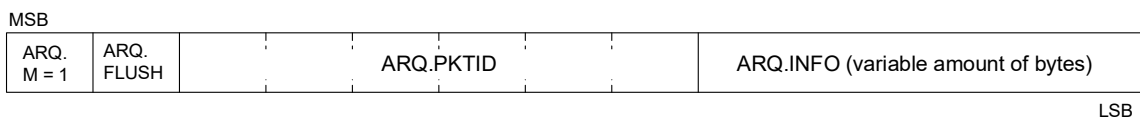


Figure 124 - ARQ subheader with ARQ.INFO

Figure 124 has the M bit in the first byte of the ARQ subheader set, and so the subheader contains multiple bytes. The first byte contains the packet ID of the transmitted packet and then follows the ARQ.INFO which is a list of one or more bytes, where each byte could have one of the following meanings:

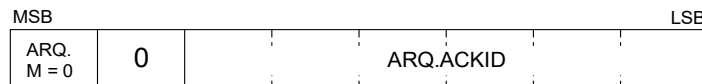


Figure 125 - ARQ.ACK byte fields

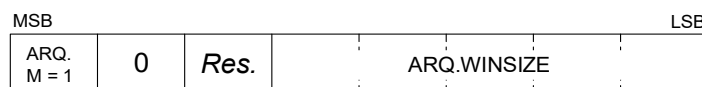


Figure 126 - ARQ.WIN byte fields

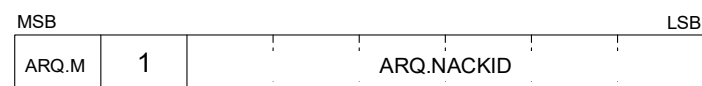


Figure 127 - ARQ.NACK byte fields

3879 If there are multiple packets lost, an ARQ.NACK is sent for each of them, from the first packet lost to the last  
 3880 packet lost. When there are several ARQ.NACK they implicitly acknowledge the packets before the first  
 3881 ARQ.NACK, and the packets in between the ARQ.NACKs. If an ARQ.ACK is present, it shall be placed at the  
 3882 end of the ARQ subheader, and shall reference to an ARQ.ACKID that is later than any other ARQ.NACKID, if  
 3883 present. If there is at least an ARQ.NACK and an ARQ.ACK they also implicitly acknowledge any packet in the  
 3884 middle between the last ARQ.NACKID and the ARQ.ACK.

3885 For interoperability, a device shall be able to receive any well-formed ARQ subheader and shall process at  
 3886 least the first ARQ.ACK or ARQ.NACK field.

3887 The subfields have the following meanings as described in Table 63

3888 **Table 63 - ARQ fields**

| Field       | Description                                                                                                              |
|-------------|--------------------------------------------------------------------------------------------------------------------------|
| ARQ.FLUSH   | ARQ.FLUSH = 1 If an ACK must be sent immediately.<br>ARQ.FLUSH = 0 If an ACK is not needed.                              |
| ARQ.PKTID   | The id of the current packet, if the packet is empty (with no data) this is the id of the packet that will be sent next. |
| ARQ.ACKID   | The identifier with the next packet expected to be received.                                                             |
| ARQ.WINSIZE | The window size available from the last acknowledged packet. After a connection is established its window is 1.          |
| ARQ.NACKID  | Ids of the packets that need to be retransmitted.                                                                        |

#### 3889 4.7.3.2.2 ARQ subheader example

|     |               |                   |                 |               |   |     |                  |
|-----|---------------|-------------------|-----------------|---------------|---|-----|------------------|
| MSB | ARQ.<br>M = 1 | ARQ.<br>FLUSH = 1 | ARQ.PKTID = 23  | ARQ.<br>M = 1 | 0 | Res | ARQ.WINSIZE = 16 |
|     | ARQ.<br>M = 1 | 1                 | ARQ.NACKID = 45 | ARQ.<br>M = 1 | 1 |     | ARQ.NACKID = 47  |
|     | ARQ.<br>M = 1 | 1                 | ARQ.NACKID = 48 | ARQ.<br>M = 1 | 1 |     | ARQ.NACKID = 52  |
|     | ARQ.<br>M = 1 | 1                 | ARQ.NACKID = 55 | ARQ.<br>M = 1 | 1 |     | ARQ.NACKID = 56  |
|     | ARQ.<br>M = 1 | 1                 | ARQ.NACKID = 57 | ARQ.<br>M = 0 | 0 |     | ARQ.ACKID = 60   |
|     |               |                   |                 |               |   |     | LSB              |

3890  
3891

**Figure 128 - Example of an ARQ subheader with all the fields present**

3892 In this example all the ARQ subheader fields are present. To make it understandable, since both Nodes are  
 3893 both transmitters and receivers, the side receiving this header will be called A and the other side transmitting  
 3894 B. The message has the packet ID of 23 if it contains data; otherwise the next data packet to be sent has the  
 3895 packet ID of 23. Since the flush bit is set it needs to be ACKed/NACKed.

3896 B requests the retransmission of packets 45, 47, 48, 52, 55, 56 and 57. ACK = 60, so it has received packets  
 3897 <45, 46, 49, 50, 51, 53, 54, 58 and 59.

3898 The window is 16 and it has received and processed up to packet 44 (first NACK = 45), so A can send all  
 3899 packets  $\leq 60$ ; that is, as well as sending the requested retransmits, it can also send packet ID = 60.

#### 3900 4.7.3.3 Windowing

3901 A new connection between two peer devices starts with an implicit initial receiver window size of 1 and a  
 3902 packet identifier 0. This window size is a limiting case and the transaction (to start with) shall behave like a  
 3903 “Stop and Wait” ARQ mechanism.

3904 On receipt of an ARQ.WIN, the sender would adapt its window size to *ARQ.WINSIZE*. This buffer size is  
 3905 counted from the first packet completely ACK-ed, so if there is a NACK list and then an ACK the window size  
 3906 defines the number of packets from the first NACK-ed packet that could be sent. If there is just an ACK in the  
 3907 packet (without any NACK) the window size determines the number of packets that can be sent from that  
 3908 ACK.

3909 An *ARQ.WINSIZE* value of 0 may be transmitted back by the receiver to indicate congestion at its end. In such  
 3910 cases, the transmitting end should wait for at least *ARQCongClrTime* before re-transmitting its data.

#### 3911 4.7.3.4 Flow control

3912 The transmitter must manage the ACK sending algorithm by the flush bit; it is up to it having a proper ARQ  
 3913 communication. The receiver is only forced to send ACKs when the transmitter has sent a packet with the  
 3914 flush bit set, although the receiver could send more ACKs even if not forced to do it, because the flow control  
 3915 is only a responsibility of the transmitter. The transmitter shall close the connection latest if the Packet  
 3916 acknowledgement is missing after *ARQMaxTxCount* Packet retransmissions. The transmitter may choose to  
 3917 use lower maximum retransmit value than *ARQMaxTxCount* and it may also close the connection any time  
 3918 earlier if it determines proper data exchange cannot be restored.

3919 These are the requisites to be interoperable, but the algorithm is up to the manufacturer. It is strongly  
 3920 recommended to piggyback data-ACK information in outgoing packets, to avoid the transmission of  
 3921 unnecessary packets just for ACK-ing. In particular in order to allow consolidated ACKs or piggybacking, the  
 3922 maximum time for each implementation before sending an ACK is *ARQMaxAckHoldTime*.

#### 3923 4.7.3.5 Algorithm recommendation

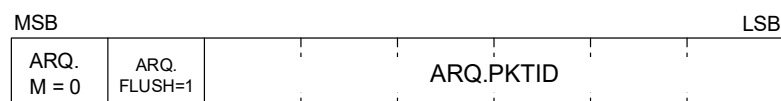
3924 No normative algorithm is specified, for a recommendation see Annex I.

#### 3925 4.7.3.6 Usage of ARQ in resource limited devices

3926 Resource limited devices may have a low memory and simple implementation of ARQ. They may want to use  
 3927 a window of 1 packet. They work as a “Stop and Wait” mechanism.

3928 The ARQ subheader to be generated shall be one of the followings:

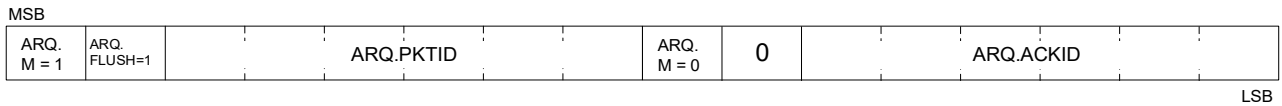
3929 If there is nothing to acknowledge:



3930  
 3931 **Figure 129 - Stop and wait ARQ subheader with only packet ID**

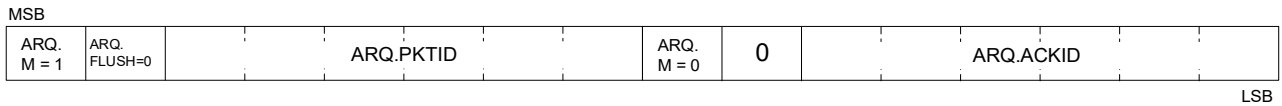


3932 If there is something to acknowledge carrying data:



3933  
3934 **Figure 130 - Stop and wait ARQ subheader with an ACK**

3935 If there is something to acknowledge but without any data in the packet:



3936  
3937 **Figure 131 - Stop and wait ARQ subheader without data and with an ACK**

3938 The ARQ.WINSIZE is not generally transmitted because the window size is already 1 by default, it only may  
3939 be transmitted to handle congestion and to resume the transmission again.

#### 3940 4.7.4 ARQ packets switching

3941 All Switch Nodes shall support transparent bridging of ARQ traffic, whether or not they support ARQ for their  
3942 own transmission and reception. In this mode, Switch Nodes are not required to buffer the packets of the  
3943 ARQ connections for retransmission.

3944 Some Switch Nodes may buffer the packets of the ARQ connections, and perform retransmission in  
3945 response to NACKs for these packets if the subnetwork is working with security profile 0. If the subnetwork  
3946 is working with security profile 1 or 2, the switch node shall not perform retransmission in response to NACKs  
3947 for these packets. The following general principles shall be followed.

- 3948 • The acknowledged packet identifiers shall have end-to-end coherency.
- 3949
- 3950 • The buffering of packets in Switch Nodes and their retransmissions shall be transparent to the source  
3951 and Destination Nodes, i.e., a Source or Destination Node shall not be required to know whether or  
3952 not an intermediate Switch has buffered packets for switched data.

#### 3953 4.8 Time Reference

3954 Packets in PRIME may interchange time references by providing a TREF subheader. Due to the frame and  
3955 superframe structure of the MAC layer, when a node is registered to a PRIME subnetwork it is already  
3956 synchronized. The TREF subheader includes a time reference that is relative to the beginning of a frame in  
3957 order to make reference to a specific moment in time.

3958 **Table 64 - Time Reference subheader fields**

| Name     | Length | Description                                                                             |
|----------|--------|-----------------------------------------------------------------------------------------|
| TREF.SEQ | 5 bits | Sequence number of the MAC Frame that is used as reference time of the event to notify. |
| Reserved | 3 bits | Always 0 for this version of the specification. Reserved for future use.                |



|           |         |                                                                                                                                                                                                                                                                                                               |
|-----------|---------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| TREF.TIME | 32 bits | Signed number in 10s of microseconds between the moment of the event, and the beginning of the frame. Positive for events after the beginning of the MAC frame, and negative for events before the beginning of the MAC frame. 0x80000000 is a special value that means that it is an invalid time reference. |
|-----------|---------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|

3959 During the transmission of a new packet, the transmitter of a TREF subheader should always keep the  
3960 TREF.SEQ as updated as possible.

3961 During the switching of a packet with a TREF subheader, the Switch Node may update the TREF.TIME and  
3962 TREF.SEQ to change the MAC frame this reference is based on, as long as it makes reference to the same  
3963 instant in time. A switch shall update the fields if  $(RX\_SEQ - TREF.SEQ) \& 31 > (TX\_SEQ - TREF.SEQ) \& 31$ ,  
3964 where RX\_SEQ is the frame sequence number when the packet is received by the switch and TX\_SEQ is the  
3965 sequence number when a packet is transmitted by the switch. In this case, this field may be easily updated  
3966 by subtracting the length of a superframe to the TREF.TIME field, leaving the same TREF.SEQ. This mechanism  
3967 is in order to avoid a superframe overlapping.

3968 If the time reference cannot be represented in the TREF.TIME field, then the field TREF.TIME should have the  
3969 value 0x80000000 that means that it is an invalid reference

## 3970 4.9 Backward Compatibility with PRIME 1.3.6

3971 In order to interoperate with Service Nodes conforming to v1.3.6 of specifications, v1.4 conformant devices  
3972 can implement a backward-compatibility mode. This mode is optional for PLC-only devices, while it is not  
3973 supported by PLC+RF and RF-only devices (PLC+RF and RF-only devices do not support this section and Annex  
3974 K and indicate REG.CAP\_BC=0, PRO.PN\_BC=0, PNH.CAP\_BC=0). As a consequence, a network including  
3975 PLC+RF and/or RF-only devices shall use the normal 1.4 MAC mode described in this specification  
3976 (BCN.MACBC=0, BCN.CSMA=0). In what follows, the MAC backward-compatibility mode, that can be  
3977 optionally supported in PRIME 1.4 networks exclusively including PLC-only devices, is described.

3978 Since PRIME v1.3.6 Service Nodes will not understand v1.4 message formats, v1.4 compliant devices  
3979 implementing backward-compatibility mode shall support additional messaging capabilities that enable  
3980 them to communicate with PRIME v1.3.6 devices. Any Subnetwork that allows registration of one or more  
3981 PRIME v1.3.6 device/s shall be termed to be running in “backward-compatibility” mode and will operate with  
3982 the following characteristics:

- 3983 • Base Node shall always be a v1.4 compliant implementation i.e. a PRIME v1.3.6 Base Node is  
3984 incapable of managing a Subnetwork in backward-compatibility mode.
- 3985 • All robust mode PDUs shall be transmitted using PHY BC Frames as defined in Annex K.
- 3986 • To accommodate for size restrictions in PHY BC Frames, the Base Node shall limit the maximum SAR  
3987 segment size to be less or equal than 64 bytes for all service nodes located, directly or indirectly,  
3988 behind a robust link

### 3989 4.9.1 Frame Structure and Channel Access

3990 A Subnetwork in “backward-compatibility” mode shall abide by principles laid down in points below:  
3991  
3992

- Fixed frame length of 276 symbols shall be used. The frames include up to five consecutive non-robust beacon slots, each of them having a length of 11.008ms.
- Transition to longer frames is prohibited. The base node transmits a beacon using DBPSK\_CC modulation in every frame in beacon slot 0. The frame format is shown in Figure 132.

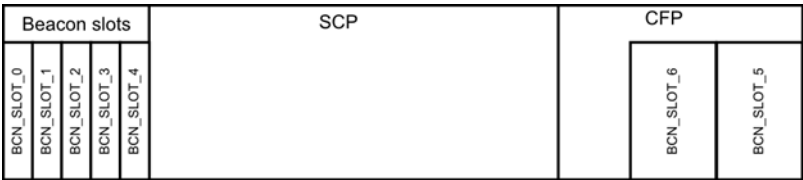


Figure 132 - CBCN Frame format for backwards compatibility mode

- PRIME v1.4 compatibility mode allows the Base Node to place up to two robust mode beacons per frame at the end of the CBCN Frame. They are located at the end of CFP, respecting the guard times (macGuardTime) before each one and at the end of the frame, as shown in the Figure 133.
- Robust beacon allocation policy is up to the manufacturer, it could be 0 to 2 robust beacons and can be allocated from the beginning or on demand.
- The robust beacons shall be sent in DBPSK\_R encoding with the PHY frame type BC.

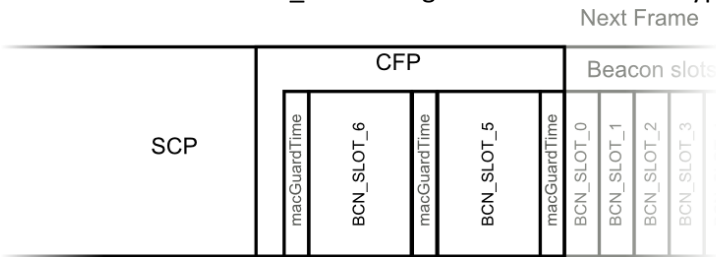


Figure 133 - Robust beacon allocation in 1.4 BC

- The Base Node shall set the CFP duration to a value which guarantees that the robust mode beacons are fully located within CBCN.CFP.
- For a Switch Node using DBPSK\_CC for beacon transmission, the Base Node allocates space from the non-robust beacon slots (slot 0 to 4). Beacon slot allocation rules according to PRIME v1.3.6 shall apply for allocation of non-robust beacon slots.
- For allocation of robust beacons the Base Node should not update the beacon slot count, as the robust beacons shall be placed in the CFP.
- The Base Node can also transmit robust beacons but it is not mandated to do so.
- If a switch or Base Node transmits robust beacons, it shall transmit at least once every 32 sub-frames.
- Frame format shall respect the same restrictions on SCP and CFP durations defined in PRIME v1.3.6.
- CSMA/CA algorithm defined in 1.3.6 shall be used.

## 4022 **4.9.2 Switching**

4023 In a network running in PRIME v1.4 compatibility mode, uplink (HDR.DO=0) Multicast and Broadcast packets  
4024 shall follow the same rules as the ones described in PRIME version 1.3.6:

- 4025 • The node shall only switch the packet that has a broadcast or multicast destination (PKT.LNID = 0x3FFF  
4026 or 0x3FFE) and that was transmitted by a Node registered through this Switch Node (PKT.SID=LSID of  
4027 this Switch Node).
- 4028 • In case a broadcast or multicast packet is switched up, the Switch Node shall replace the PKT.SID with  
4029 Switch Node's SID.

4030 For the rest of the packet types the Switch Node shall follow the rules and actions described in chapter 4.3.5.

## 4031 **4.9.3 PDU Frame Formats**

### 4032 **4.9.3.1 General Format**

4033 In a network running in PRIME v1.4 compatibility mode, all nodes shall use the standard Generic Mac  
4034 Header (see Section 4.4.2.4) and the Compatibility Packet Header (CPKT, see Annex K). The CRC calculation  
4035 follows the standard procedure described in Section 4.4.2. These headers and CRC calculation follow the  
4036 PRIME v1.3.6 specification.

4037 In a compatibility mode network, some control messages need a different format from the standard PRIME  
4038 v1.4 format. This is for example the case for messages which are sniffed by PRIME v1.3.6 devices. These  
4039 special messages are listed in the following sub-sections. On the other hand, the standard PRIME v1.4  
4040 payload format is used for the CON, CFP, MUL and SEC control packets.

### 4041 **4.9.3.2 Registration and Unregistration control messages**

4042 A mixture of compatibility mode registration messages (CREG, Annex K.1.1.1.1), which follow the following  
4043 PRIME v1.3.6 message format, and PRIME v1.4 format REG control messages shall be used. For a detailed  
4044 description of the registration procedure in a compatibility mode network see Annex K.2.1.

4045 The messages used during unregistration shall follow the CREG frame format specified in Annex K.1.1.1.1.

### 4046 **4.9.3.3 Promotion and Beacon Slot Indication control messages**

4047 For all promotion messages the compatibility mode format CPRO (see Annex K.1.1.1.2) shall be used. The  
4048 CREG messages resemble PRIME v1.3.6 messages. This is important as switches on the branch need to sniff  
4049 these packets in order to refresh their switch tables. The compatibility mode promotion procedure, which is  
4050 described in Annex K.2.3, requires also compatibility BSI packets (CBSI). The BSI packets are no longer used  
4051 in PRIME v1.4. A compatibility mode network does not support a modulation change of a switch.

4052 The messages used during demotion shall follow the CPRO frame format specified in Annex K.1.1.1.2.

### 4053 **4.9.3.4 Keep-Alive control messages**

4054 PRIME v1.3.6 service nodes do not know about the new link level keep-alive process. Therefore, for networks  
4055 operating in backward compatibility mode, the keep-alive process shall remain the same as in PRIME v1.3.6.  
4056 The process is described in Annex K.2.5. In addition, the CALV control messages are also enumerated in  
4057 K.1.1.1.5.

## 5 Convergence layer

### 5.1 Overview

Figure 134 shows the overall structure of the Convergence layer.

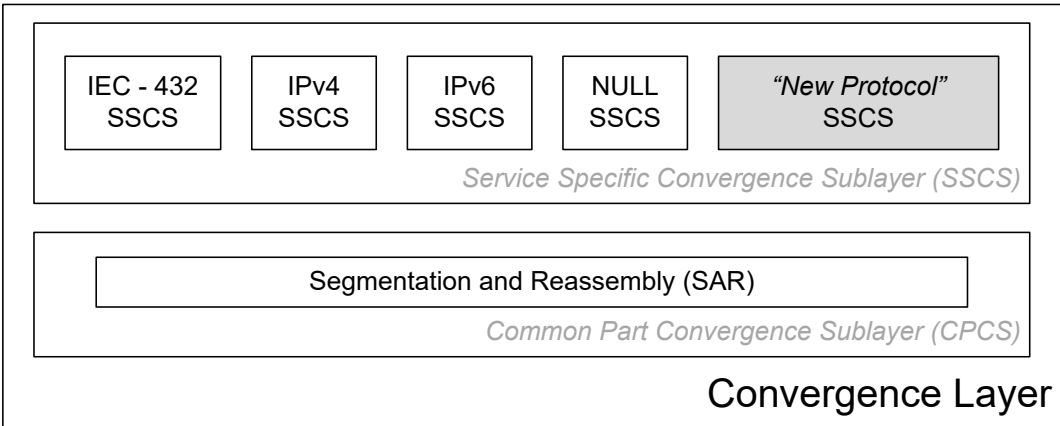


Figure 134 - Structure of the Convergence layer

The Convergence layer is separated into two sublayers. The Common Part Convergence Sublayer (CPCS) provides a set of generic services. The Service Specific Convergence Sublayer (SSCS) contains services that are specific to one communication profile. There are several SSCSs, typically one per communication profile, but only one CPCS. The use of CPCS services is optional in that a certain SSCS will use the services it needs from the CPCS, and omit services which are not needed.

### 5.2 Common Part Convergence Sublayer (CPCS)

#### 5.2.1 General

This specification defines only one CPCS service: Segmentation and Reassembly (SAR).

#### 5.2.2 Segmentation and Reassembly (SAR)

##### 5.2.2.1 General

CPCS SDUs which are larger than 'macSARSize-1' bytes are segmented at the CPCS. CPCS SDUs which are equal or smaller than 'macSARSize -1' bytes may also optionally be segmented. Segmentation means breaking up a CPCS SDU into smaller parts to be transferred by the MAC layer. At the peer CPCS, the smaller parts (segments) are put back together (i.e. reassembled) to form the complete CPCS SDU. All segments except the last segment of a segmented SDU must be the same size and at most macSARSize bytes in length. Segments may be decided to be smaller than 'macSARSize -1' bytes e.g. when the channel is poor. The last segment may of course be smaller than 'macSARSize -1' bytes.

In order to keep SAR functionality simple, the macSARSize is a constant value for all possible modulation/coding combinations at PHY layer. The value of macSARSize is such that with any modulation/coding combination, it is always possible to transmit a single segment in one PPDU. Therefore, there is no need for discovering a specific MTU between peer CPCSs or modifying the SAR configuration for

every change in the modulation/coding combination. In order to increase efficiency, a Service Node which supports packet aggregation may combine multiple segments into one PPDU when communicating with its peer.

Segmentation always adds a 1-byte header to each segment. The first 2 bits of SAR header identify the type of segment. The semantics of the rest of the header information then depend on the type of segment. The structure of different header types is shown in Figure 135 and individual fields are explained in Table 65. Not all fields are present in each SAR header. Either SAR.NSEGS or SAR.SEQ is present, but not both.

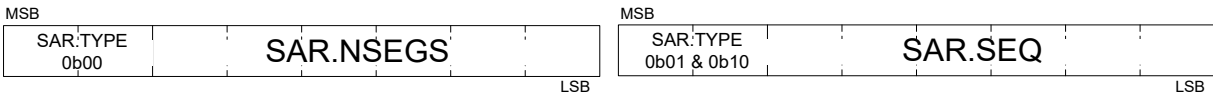


Figure 135 – Segmentation and Reassembly Headers

Table 65 - SAR header fields

| Name      | Length | Description                                                                                                                                                                                                                            |
|-----------|--------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| SAR.TYPE  | 2 bits | Type of segment. <ul style="list-style-type: none"><li>• 0b00: first segment;</li><li>• 0b01: intermediate segment;</li><li>• 0b10: last segment;</li><li>• 0b11: Last segment with SAR.CRC field at the end of the segment.</li></ul> |
| SAR.NSEGS | 6 bits | ‘Number of Segments’ – 1.<br><br>Note: This field is only present in segments with SAR.TYPE=0b00                                                                                                                                       |
| SAR.SEQ   | 6 bits | Sequence number of segment.<br><br>Note: This field is only present in segments with SAR.TYPE=0b01, SAR.TYPE=0b10 and SAR.TYPE=0b11.                                                                                                   |

Every segment (except for the first one) includes a sequence number so that the loss of a segment could be detected in reassembly. The sequence numbering shall start from zero with every new CPCS SDU. The first segment which contains a SAR.SEQ field must have SAR.SEQ = 0. All subsequent segments from the same CPCS SDU shall increase this sequence number such that the SAR.SEQ field adds one with every transmission.

The value SAR.NSEGS indicates the total number of segments, minus one. So when SAR.NSEGS = 0, the CPCS SDU is sent in one segment. SAR.NSEGS = 63 indicates there will be 64 segments to form the full CPCS SDU. When SAR.NSEGS = 0, it indicates that this first segment is also the last segment. No further segment with SAR.TYPE = 0b01 or 0b10 is to be expected for this one-segment CPCS SDU.

Using segments with SAR.TYPE=0b11 instead of SAR.TYPE=0b10 will be recommended for the last segments of connections without ARQ, multicast and broadcast. Connections with ARQ may use SAR.TYPE=0b10 safely (this reduces overhead and guarantees backward compatibility). The 32 bits CRC is computed using the polynomial generator in the and appended at the end of the last segment.

Table 66 – SAR.CRC

| Name    | Length  | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |
|---------|---------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| SAR.CRC | 32 bits | <p>CRC32 of the SAR CPCS PDU.</p> <p>This field is only present in segments with SAR.TYPE=0b11.</p> <p>The input polynomial <math>M(x)</math> is formed as a polynomial whose coefficients are bits of the data being checked (the first bit to check is the highest order coefficient and the last bit to check is the coefficient of order zero). The Generator polynomial for the CRC is <math>G(x)=x^{32}+x^{26}+x^{23}+x^{22}+x^{16}+x^{12}+x^{11}+x^{10}+x^8+x^7+x^5+x^4+x^2+x+1</math>. The remainder <math>R(x)</math> is calculated as the remainder from the division of <math>M(x) \cdot x^{32}</math> by <math>G(x)</math>. The coefficients of the remainder will then be the resulting CRC.</p> |

4108

4109 SAR at the receiving end shall buffer all segments and deliver only fully reassembled CPCS SDUs to the SSCS  
4110 above. Should reassembly fail due to a segment not being received or too many segments being ...received  
4111 etc., SAR shall not deliver any incomplete CPCS SDU to the SSCS above.

4112 **5.2.2.2 SAR constants**

4113 Table shows the constants for the SAR service.

4114 **Table 67 - SAR Constants**

| Constant        | Value                               |
|-----------------|-------------------------------------|
| CIMaxAppPktSize | Max Value (SAR.NSEGS) x macSARSize. |

4115 **5.3 NULL Service-Specific Convergence Sublayer (NULL SSCS)**

4116 **5.3.1 Overview**

4117 Null SSCS provides the MAC layer with a transparent path to upper layers, being as simple as possible and  
4118 minimizing overhead. It is intended for applications that do not need any special convergence capability.

4119 The unicast and multicast connections of this SSCS shall use the SAR service, as defined in 5.2.2. If they do  
4120 not need the SAR service they shall still include the SAR header (notifying just one segment).

4121 The CON.TYPE and MUL.TYPE (see Annex E) for unicast connections and multicast groups shall use the same  
4122 type that has been already defined for the application that makes use of this Null SSCS.

4123 **5.3.2 Primitives**

4124 Null SSCS primitives are just a direct mapping of the MAC primitives. A full description of every primitive is  
4125 avoided, because the mapping is direct and they will work as the ones of the MAC layer.

4126 The directly mapped primitives have exactly the same parameters as the ones in the MAC layer and perform  
4127 the same functionality. The set of primitives that are directly mapped are shown below.

Table 68 - Primitive mapping between the Null SSCS primitives and the MAC layer primitives

| Null SSCS mapped to ...      | ... a MAC primitive      |
|------------------------------|--------------------------|
| CL_NULL_ESTABLISH.request    | MAC_ESTABLISH.request    |
| CL_NULL_ESTABLISH.indication | MAC_ESTABLISH.indication |
| CL_NULL_ESTABLISH.response   | MAC_ESTABLISH.response   |
| CL_NULL_ESTABLISH.confirm    | MAC_ESTABLISH.confirm    |
| CL_NULL_RELEASE.request      | MAC_RELEASE.request      |
| CL_NULL_RELEASE.indication   | MAC_RELEASE.indication   |
| CL_NULL_RELEASE.response     | MAC_RELEASE.response     |
| CL_NULL_RELEASE.confirm      | MAC_RELEASE.confirm      |
| CL_NULL_JOIN.request         | MAC_JOIN.request         |
| CL_NULL_JOIN.indication      | MAC_JOIN.indication      |
| CL_NULL_JOIN.response        | MAC_JOIN.response        |
| CL_NULL_JOIN.confirm         | MAC_JOIN.confirm         |
| CL_NULL_LEAVE.request        | MAC_LEAVE.request        |
| CL_NULL_LEAVE.indication     | MAC_LEAVE.indication     |
| CL_NULL_LEAVE.response       | MAC_LEAVE.response       |
| CL_NULL_LEAVE.confirm        | MAC_LEAVE.confirm        |
| CL_NULL_DATA.request         | MAC_DATA.request         |
| CL_NULL_DATA.indication      | MAC_DATA.indication      |
| CL_NULL_DATA.confirm         | MAC_DATA.confirm         |

## 5.4 IPv4 Service-Specific Convergence Sublayer (IPv4 SSCS)

### 5.4.1 Overview

The IPv4 SSCS provides an efficient method for transferring IPv4 packets over the PRIME Subnetworks. Several conventions do apply:

- A Service Node can send IPv4 packets to the Base Node or to other Service Nodes.
- It is assumed that the Base Node acts as a router between the PRIME Subnetwork and any other network. The Base Node could also act as a NAT. How the Base Node connects to the other networks is beyond the scope of this specification.
- In order to keep implementations simple, only one single route is supported per local IPv4 address.
- Service Nodes may use statically configured IPv4 addresses or DHCP to obtain IPv4 addresses.
- The Base Node performs IPv4 to EUI-48 address resolution. Each Service Node registers its IPv4 address and EUI-48 address with the Base Node (see section 5.4.2). Other Service Nodes can then query the Base Node to resolve an IPv4 address into a EUI-48 address. This requires the establishment of a dedicated connection with the Base Node for address resolution.
- The IPv4 SSCS performs the routing of IPv4 packets. In other words, the IPv4 SSCS will decide whether the packet should be sent directly to another Service Node or forwarded to the configured gateway.



- Although IPv4 is a connectionless protocol, the IPv4 SSCS is connection-oriented. Once address resolution has been performed, a connection is established between the source and destination Service Node for the transfer of IPv4 packets. This connection is maintained while traffic is being transferred and may be closed after a period of inactivity.
- The CPCS (see section 5.2) SAR sublayer shall always be present with the IPv4 Convergence layer. Generated MSDUs are at most ‘macSARSize’ bytes long and upper layer PDU messages are not expected must not to be longer than CIMaxAppPktSize.
- Optionally TCP/IPv4 headers may be compressed. Compression is negotiated as part of the connection establishment phase.
- The broadcasting of IPv4 packets is supported using the MAC broadcast mechanism.
- The multicasting of IPv4 packets is supported using the MAC multicast mechanism.

The IPv4 SSCS has a number of connection types. For address resolution there is a connection to the Base Node. For IPv4 data transfer there is one connection per Destination Node: with the Base Node that acts as the IPv4 gateway to other networks or to/with any other Node in the same Subnetwork. This is shown in Figure 136.

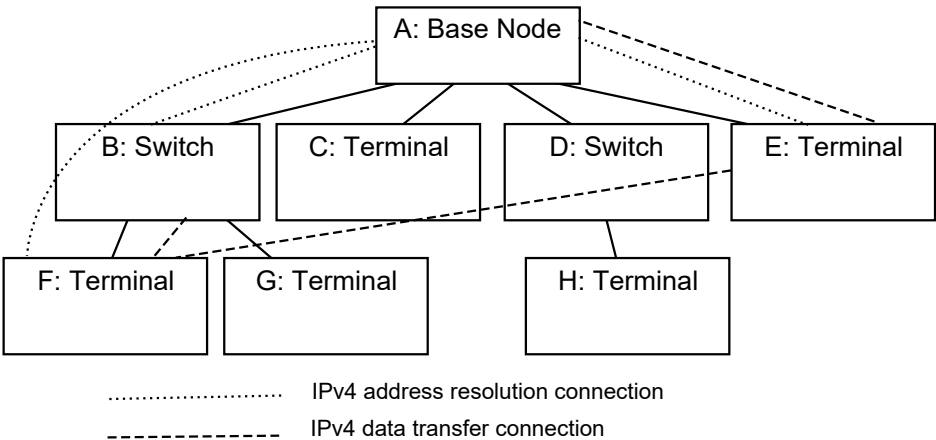


Figure 136 - IPv4 SSCS connection example

Here, Nodes B, E and F have address resolution connections to the Base Node. Node E has a data connection to the Base Node and Node F. Node F is also has a data connection to Node B. The figure does not show broadcast and multicast connections.

## 5.4.2 Address resolution

### 5.4.2.1 General

The IPv4 layer will present the IPV4 SSCS with an IPV4 packet to be transferred. The IPV4 SSCS is responsible for determining which Service Node the packet should be delivered to using the IPV4 addresses in the packet. The IPV4 SSCS must then establish a connection to the destination if one does not already exist so that the packet can be transferred. Three classes of IPV4 addresses can be used and the following subsections describe how these addresses are resolved into EUI-48 addresses.



## 4175 **5.4.2.2 Unicast addresses**

### 4176 **5.4.2.2.1 General**

4177 IPv4 unicast addresses must be resolved into unicast EUI-48 addresses. The Base Node maintains a database  
4178 of IPv4 addresses and EUI-48 addresses. Address resolution then operates by querying this database. A  
4179 Service Node must establish a connection to the address resolution service running on the Base Node, using  
4180 the connection type value TYPE (see Annex E) TYPE\_CL\_IPv4\_AR. No data should be passed in the connection  
4181 establishment. Using this connection, the Service Node can use two mechanisms as defined in the following  
4182 paragraphs.

### 4183 **5.4.2.2.2 Address registration and unregistration**

4184 A Service Node uses the AR\_REGISTER\_S message to register an IPv4 address and the corresponding EUI-48  
4185 address meaning request from the base node to record inside its registration table, the IPv4 address and its  
4186 corresponding service node EUI-48. The Base Node will acknowledge an AR\_REGISTER\_B message. The  
4187 Service Node may register multiple IPv4 addresses for the same EUI-48 address.

4188 A Service Node uses the AR\_DEREGISTER\_S message to unregister an IPv4 address and the corresponding  
4189 EUI-48 address meaning requests from the base node to delete inside its registration table, the entry  
4190 corresponding to the concerned IPv4 address. The Base Node will acknowledge it with an AR\_DEREGISTER\_B  
4191 message.

4192 When the IPv4 address resolution connection between the Service Node and the Base Node is closed, the  
4193 Base Node should remove all addresses associated to that connection.

### 4194 **5.4.2.2.3 Address lookup**

4195 A Service Node uses the AR\_LOOKUP\_S message to perform a lookup. The message contains the IPv4 address  
4196 to be resolved. The Base Node will respond with an AR\_LOOKUP\_B message that contains an error code and,  
4197 if there is no error, the EUI-48 address associated with the IPv4 address. If the Base Node has multiple entries  
4198 in its database for the same IPv4 address, the possible returned EUI-48 address is undefined.

### 4199 **5.4.2.3 Broadcast Address**

4200 IPv4 broadcast address 255.255.255.255 maps to a MAC broadcast connection with LCID equal to  
4201 LCI\_CL\_IPv4\_BROADCAST. All IPv4 broadcast packets will be sent to this connection. When an IPv4 broadcast  
4202 packet is received on this connection, the IPv4 address should be examined to determine if it is a broadcast  
4203 packet for the Subnetwork in which the Node has an IPv4 address. Only broadcast packets from member  
4204 subnets should be passed up the IPv4 protocol stack.

### 4205 **5.4.2.4 Multicast Addresses**

4206 Multicast IPv4 addresses are mapped to a PRIME MAC multicast connection by the Base Node using an  
4207 address resolution protocol.

4208 To join a multicast group, AR\_MCAST\_REG\_S is sent from the Service Node to the Base Node with the IPv4  
4209 multicast address. The Base Node will reply with an AR\_MCAST\_REG\_B that contains the LCID value assigned  
4210 to the said multicast address. However, the Base Node may also allocate other LCIDs which are not in use if  
4211 it so wishes. The Service Node can then join a multicast group (see 4.6.7.2) for the given LCID to receive IPv4

4212 multicast packets. These LCID values can be reused so that multiple IPv4 destination multicast addresses can  
 4213 be seen on the same LCID. To leave the multicast group, AR\_MCAST\_UNREG\_S is sent from the Service Node  
 4214 to the Base Node with the IPv4 multicast address. The Base Node will acknowledge it with an  
 4215 AR\_MCAST\_UNREG\_B message.

4216 When a Service Node wants to send an IPv4 multicast datagram, it just uses the appropriate LCID. If the  
 4217 Service Node has not joined the multicast group, it needs first to learn the LCID to be used. The process with  
 4218 AR\_MCAST\_REG\_{S|B} messages as described above can be used. While IPv4 multicast packets are still being  
 4219 sent, the Service Node remains registered to the multicast group.  $T_{mcast\_reg}$  after the last IPv4 multicast  
 4220 datagram was sent, the Service Node should unregister from the multicast group, by means of  
 4221 AR\_MCAST\_UNREG\_{S|B} messages. The nominal value of  $T_{mcast\_reg}$  is 10 minutes; however, other values  
 4222 may be used.

#### 4223 5.4.2.5 Retransmission of address resolution packets

4224 The connection between the Service Node and the Base Node for address resolution is not reliable if the MAC  
 4225 ARQ is not used. The Service Node is responsible for making retransmissions if the Base Node does not  
 4226 respond in one second. It is not considered an error when the Base Node receives the same registration  
 4227 requests multiple times or is asked to remove a registration that does not exist. These conditions can be the  
 4228 result of retransmissions.

#### 4229 5.4.3 IPv4 packet transfer

4230 For packets to be transferred, a connection needs to be established between source and Destination Nodes.  
 4231 The IPV4 SSCS will examine each IPv4 packet to determine the destination EUI-48 address. If a data  
 4232 connection to the destination already exists, the packet is sent. To establish this, IPv4 SSCS keeps a table for  
 4233 each connection, with information shown in Table 69 (see RFC 1144).. To use this table, it is first necessary  
 4234 to determine if the IPv4 destination address is in the local Subnetwork or if a gateway has to be used. The  
 4235 netmask associated with the local IPv4 address is used to determine this. If the IPv4 destination address is  
 4236 not in the local Subnetwork, the address of the default gateway is used instead of the destination address  
 4237 when the table is searched.

4238 **Table 69 - IPV4 SSCS Table Entry**

| Parameter             | Description                                     |
|-----------------------|-------------------------------------------------|
| CL_IPv4_Con.Remote_IP | Remote IPv4 address of this connection.         |
| CL_IPv4_Con.ConHandle | MAC Connection handle for the connection.       |
| CL_IPv4_Con.LastUsed  | Timestamp of last packet received/transmitted . |
| CL_IPv4_Con.HC        | Header Compression scheme being used.           |
| CL_IPv4_CON.RxSeq     | Next expected Receive sequence number.          |
| CL_IPv4_CON.TxSeq     | Sequence number for next transmission.          |

4239 The IPV4 SSCS may close a connection when it has not been used for an implementation-defined time period.  
4240 When the connection is closed the entry for the connection is removed at both ends of the connection.

4241 When a connection to the destination does not exist, more work is necessary. The address resolution service  
4242 is used to determine the EUI-48 address of the remote IPv4 address if it is local or the gateway associated  
4243 with the local address if the destination address is in another Subnetwork. When the Base Node replies with  
4244 the EUI-48 address of the destination Service Node, a MAC connection is established to the remote device.  
4245 The TYPE value of this connection is TYPE\_CL\_IPv4\_UNICAST. The data passed in the request message is  
4246 defined in section 5.4.7.4. The local IPv4 address is provided so that the remote device can add the new  
4247 connection to its cache of connections for sending data in the opposite direction. The use of Van Jacobson  
4248 Header Compression is also negotiated as part of the connection establishment. Once the connection has  
4249 been established, the IPv4 packet can be sent.

4250 When the packet is addressed to the IPv4 broadcast address, the packet has to be sent using the MAC  
4251 broadcast service. When the IPV4 SSCS is opened, a broadcast connection is established for transferring all  
4252 broadcast packets. The broadcast IPv4 packet is simply sent to this connection. Any packet received on this  
4253 broadcast connection is passed to the IPv4 protocol stack.

#### 4254 **5.4.4 Segmentation and reassembly**

4255 The IPV4 SSCS should support IPv4 packets with an MTU of 1500 bytes. This requires the use of SAR (see  
4256 5.2.2).

#### 4257 **5.4.5 Header compression**

4258 Van Jacobson TCP/IP Header Compression is an optional feature in the IPv4 SSCS. The use of VJ compression  
4259 is negotiated as part of the connection establishment phase of the connection between two Service Nodes.

4260 VJ compression is designed for use over a point-to-point link layer that can inform the decompressor when  
4261 packets have been corrupted or lost. When there are errors or lost packets, the decompressor can then  
4262 resynchronize with the compressor. Without this resynchronization process, erroneous packets will be  
4263 produced and passed up the IPv4 stack.

4264 The MAC layer does not provide the facility of detecting lost packets or reporting corrupt packets. Thus, it is  
4265 necessary to add this functionality in the IPV4 SSCS. The IPV4 SSCS maintains two sequence numbers when  
4266 VJ compression is enabled for a connection. These sequence numbers are 8 bits in size. When transmitting  
4267 an IPv4 packet, the CL\_IPv4\_CON.TxSeq sequence number is placed in the packet header, as shown in Section  
4268 5.4.3. The sequence number is then incremented. Upon reception of a packet, the sequence number in the  
4269 received packet is compared against CL\_IPv4\_CON.RxSeq. If they differ, TYPE\_ERROR, as defined in RFC1144,  
4270 is passed to the decompressor. The CL\_IPv4\_CON.RxSeq value is always updated to the value received in the  
4271 packet header.

4272 Header compression should never be negotiated for broadcast or multicast packets.

#### 4273 **5.4.6 Quality of Service mapping**

4274 The PRIME MAC specifies that the contention-based access mechanism supports 4 priority levels (1-4). Level  
4275 1 is used for MAC control messages, but not exclusively so.

IPv4 packets include a TOS field in the header to indicate the QoS the packet would like to receive. Three bits of the TOS indicate the IP Precedence. The following table specifies how the IP Precedence is mapped into the PRIME MAC priority.

**Table 70 - Mapping IPv4 Precedence to PRIME MAC priority**

| IP Precedence              | MAC Priority |
|----------------------------|--------------|
| 000 – Routine              | 3            |
| 001 – Priority             | 3            |
| 010 – Immediate            | 2            |
| 011 – Flash                | 2            |
| 100 – Flash Override       | 1            |
| 101 – Critical             | 1            |
| 110 – Internetwork Control | 0            |
| 111 – Network Control      | 0            |

## 5.4.7 Packet formats and connection data

### 5.4.7.1 General

This section defines the format of IPV4 SCS PDUs.

### 5.4.7.2 Address resolution PDUs

#### 5.4.7.2.1 General

The following PDUs are transferred over the address resolution connection between the Service Node and the Base Node. The following sections define AR.MSG values in the range of 0 to 11. All higher values are reserved for later versions of this specification.

#### 5.4.7.2.2 AR\_REGISTER\_S

Table 71 shows the address resolution register message sent from the Service Node to the Base Node.

**Table 71 - AR\_REGISTER\_S message format**

| Name    | Length  | Description                                                                                             |
|---------|---------|---------------------------------------------------------------------------------------------------------|
| AR.MSG  | 8-bits  | Address Resolution Message Type. <ul style="list-style-type: none"><li>For AR_REGISTER_S = 0.</li></ul> |
| AR.IPv4 | 32-bits | IPv4 address to be registered.                                                                          |

|           |         |                          |
|-----------|---------|--------------------------|
| AR.EUI-48 | 48-bits | EUI-48 to be registered. |
|-----------|---------|--------------------------|

#### 4291 5.4.7.2.3 AR\_REGISTER\_B

4292 Table 72 shows the address resolution register acknowledgment message sent from the Base Node to the  
4293 Service Node.

4294 Table 72 - AR\_REGISTER\_B message format

| Name      | Length  | Description                                                                                                |
|-----------|---------|------------------------------------------------------------------------------------------------------------|
| AR.MSG    | 8-bits  | Address Resolution Message Type.<br><ul style="list-style-type: none"><li>For AR_REGISTER_B = 1.</li></ul> |
| AR.IPv4   | 32-bits | Registered IPv4 address.                                                                                   |
| AR.EUI-48 | 48-bits | EUI-48 registered.                                                                                         |

4295

4296 The AR.IPv4 and AR.EUI-48 fields are included in the AR\_REGISTER\_B message so that the Service Node can  
4297 perform multiple overlapping registrations.

#### 4298 5.4.7.2.4 AR\_UNREGISTER\_S

4299 Table 73 shows the address resolution unregister message sent from the Service Node to the Base Node.

4300 Table 73 - AR\_UNREGISTER\_S message format

| Name      | Length  | Description                                                                                                  |
|-----------|---------|--------------------------------------------------------------------------------------------------------------|
| AR.MSG    | 8-bits  | Address Resolution Message Type.<br><ul style="list-style-type: none"><li>For AR_UNREGISTER_S = 2.</li></ul> |
| AR.IPv4   | 32-bits | IPv4 address to be unregistered.                                                                             |
| AR.EUI-48 | 48-bits | EUI-48 to be unregistered.                                                                                   |

#### 4301 5.4.7.2.5 AR\_UNREGISTER\_B

4302 Table 74 shows the address resolution unregister acknowledgment message sent from the Base Node to the  
4303 Service Node.

4304 Table 74 - AR\_UNREGISTER\_B message format

| Name   | Length | Description                                                                                                  |
|--------|--------|--------------------------------------------------------------------------------------------------------------|
| AR.MSG | 8-bits | Address Resolution Message Type.<br><ul style="list-style-type: none"><li>For AR_UNREGISTER_B = 3.</li></ul> |

|           |         |                             |
|-----------|---------|-----------------------------|
| AR.IPv4   | 32-bits | Unregistered IPv4 address . |
| AR.EUI-48 | 48-bits | Unregistered EUI-48.        |

4305 The AR.IPv4 and AR.EUI-48 fields are included in the AR\_UNREGISTER\_B message so that the Service Node  
4306 can perform multiple overlapping Unregistrations.

#### 4307 5.4.7.2.6 AR\_LOOKUP\_S

4308 Table 75 shows the address resolution lookup message sent from the Service Node to the Base Node.

4309 Table 75 - AR\_LOOKUP\_S message format

| Name    | Length  | Description                                                                                             |
|---------|---------|---------------------------------------------------------------------------------------------------------|
| AR.MSG  | 8-bits  | Address Resolution Message Type. <ul style="list-style-type: none"><li>• For AR_LOOKUP_S = 4.</li></ul> |
| AR.IPv4 | 32-bits | IPv4 address to lookup.                                                                                 |

#### 4310 5.4.7.2.7 AR\_LOOKUP\_B

4311 Table 76 shows the address resolution lookup response message sent from the Base Node to the Service  
4312 Node.

4313 Table 76 - AR\_LOOKUP\_B message format

| Name      | Length  | Description                                                                                                                                                                                        |
|-----------|---------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| AR.MSG    | 8-bits  | Address Resolution Message Type. <ul style="list-style-type: none"><li>• For AR_LOOKUP_B = 5.</li></ul>                                                                                            |
| AR.IPv4   | 32-bits | IPv4 address looked up.                                                                                                                                                                            |
| AR.EUI-48 | 48-bits | EUI-48 for IPv4 address.                                                                                                                                                                           |
| AR.Status | 8-bits  | Lookup status, indicating if the address was found or an error occurred. <ul style="list-style-type: none"><li>• 0 = found, AR.EUI-48 valid;</li><li>• 1 = unknown, AR.EUI-48 undefined.</li></ul> |

4314 The lookup may fail if the requested address has not been registered. In that case, AR.Status will have a value  
4315 other than zero and the contents of AR.EUI-48 will be undefined. The lookup is only successful when  
4316 AR.Status is zero. In that case, the EUI-48 field contains the resolved address.

#### 4317 5.4.7.2.8 AR\_MCAST\_REG\_S

4318 Table 77 shows the multicast address resolution register message sent from the Service Node to the Base  
4319 Node.

4320 Table 77 - AR\_MCAST\_REG\_S message format

| Name    | Length  | Description                                                                                              |
|---------|---------|----------------------------------------------------------------------------------------------------------|
| AR.MSG  | 8-bits  | Address Resolution Message Type. <ul style="list-style-type: none"><li>For AR_MCAST_REG_S = 8.</li></ul> |
| AR.IPv4 | 32-bits | IPv4 multicast address to be registered.                                                                 |

#### 5.4.7.2.9 AR\_MCAST\_REG\_B

Table 78 shows the multicast address resolution register acknowledgment message sent from the Base Node to the Service Node.

Table 78 - AR\_MCAST\_REG\_B message format

| Name     | Length  | Description                                                                                              |
|----------|---------|----------------------------------------------------------------------------------------------------------|
| AR.MSG   | 8-bits  | Address Resolution Message Type. <ul style="list-style-type: none"><li>For AR_MCAST_REG_B = 9.</li></ul> |
| AR.IPv4  | 32-bits | IPv4 multicast address registered.                                                                       |
| Reserved | 2-bits  | Reserved. Should be encoded as 0.                                                                        |
| AR.LCID  | 6-bits  | LCID assigned to this IPv4 multicast address.                                                            |

4325

The AR.IPv4 field is included in the AR\_MCAST\_REG\_B message so that the Service Node can perform multiple overlapping registrations.

#### 5.4.7.2.10 AR\_MCAST\_UNREG\_S

Table 79 shows the multicast address resolution unregister message sent from the Service Node to the Base Node.

Table 79 - AR\_MCAST\_UNREG\_S message format

| Name    | Length  | Description                                                                                                 |
|---------|---------|-------------------------------------------------------------------------------------------------------------|
| AR.MSG  | 8-bits  | Address Resolution Message Type. <ul style="list-style-type: none"><li>For AR_MCAST_UNREG_S = 10.</li></ul> |
| AR.IPv4 | 32-bits | IPv4 multicast address to be unregistered.                                                                  |

#### 5.4.7.2.11 AR\_MCAST\_UNREG\_B

Table 80 shows the multicast address resolution unregister acknowledgment message sent from the Base Node to the Service Node.

Table 80 - AR\_MCAST\_UNREG\_B message format

| Name    | Length  | Description                                                                                                          |
|---------|---------|----------------------------------------------------------------------------------------------------------------------|
| AR.MSG  | 8-bits  | Address Resolution Message Type.<br><br><ul style="list-style-type: none"> <li>For AR_MCAST_UNREG_B = 11;</li> </ul> |
| AR.IPv4 | 32-bits | IPv4 multicast address unregistered.                                                                                 |

4336

4337 The AR.IPv4 field is included in the AR\_MCAST\_UNREG\_B message so that the Service Node can perform  
 4338 multiple overlapping Unregistrations.

### 4339 5.4.7.3 IPv4 packet format

#### 4340 5.4.7.3.1 General

4341 The following PDU formats are used for transferring IPv4 packets between Service Nodes. Two formats are  
 4342 defined. The first format is for when header compression is not used. The second format is for Van Jacobson  
 4343 Header Compression.

#### 4344 5.4.7.3.2 IPv4 Packet Format, No Negotiated Header Compression

4345 When no header compression has been negotiated, the IPv4 packet is simply sent as is, without any header.

4346 Table 81 - IPv4 Packet format without negotiated header compression

| Name            | Length   | Description      |
|-----------------|----------|------------------|
| <i>IPv4.PKT</i> | n-octets | The IPv4 Packet. |

#### 4347 5.4.7.3.3 IPv4 Packet Format with VJ Header Compression

4348 With Van Jacobsen header compression, a one-octet header is needed before the IPv4 packet.

4349 Table 82 - IPv4 Packet format with VJ header compression negotiated

| Name            | Length   | Description                                                                                                                                                                                                                           |
|-----------------|----------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| IPv4.Type       | 2-bits   | Type of compressed packet.<br><br><ul style="list-style-type: none"> <li>IPv4.Type = 0 – TYPE_IP;</li> <li>IPv4.Type = 1 – UNCOMPRESSED_TCP;</li> <li>IPv4.Type = 2 – COMPRESSED_TCP;</li> <li>IPv4.Type = 3 – TYPE_ERROR.</li> </ul> |
| IPv4.Seq        | 6-bits   | Packet sequence number.                                                                                                                                                                                                               |
| <i>IPv4.PKT</i> | n-octets | The IPv4 Packet.                                                                                                                                                                                                                      |

4350



The IPv4.Type value TYPE\_ERROR is never sent. It is a pseudo packet type used to tell the decompressor that a packet has been lost.

#### 5.4.7.4 Connection Data

##### 5.4.7.4.1 General

When a connection is established between Service Nodes for the transfer of IPv4 packets, data is also transferred in the connection request packets. This data allows the negotiation of compression and notification of the IPv4 address.

##### 5.4.7.4.2 Connection Data from the Initiator

Table 83 shows the connection data sent by the initiator.

Table 83 - Connection data sent by the initiator

| Name            | Length  | Description                                                                                                                                                                                                                                    |
|-----------------|---------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>Reserved</i> | 6-bits  | Should be encoded as 0 in this version of the IPV4 SSCS protocol.                                                                                                                                                                              |
| Data.HC         | 2-bit   | Header Compression . <ul style="list-style-type: none"><li>• Data.HC = 0 – No compression requested;</li><li>• Data.HC = 1 – VJ Compression requested;</li><li>• Data.HC = 2, 3 – Reserved for future versions of the specification.</li></ul> |
| Data.IPv4       | 32-bits | IPv4 address of the initiator                                                                                                                                                                                                                  |

If the device accepts the connection, it should copy the Data.IPv4 address into a new table entry along with the negotiated Data.HC value.

##### 5.4.7.4.3 Connection Data from the Responder

Table 84 shows the connection data sent in response to the connection request.

Table 84 - Connection data sent by the responder

| Name            | Length | Description                                                                                                                                                                                                     |
|-----------------|--------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>Reserved</i> | 6-bits | Should be encoded as zero in this version of the IPV4 SSCS protocol.                                                                                                                                            |
| Data.HC         | 2-bit  | Header Compression negotiated. <ul style="list-style-type: none"><li>• Data.HC = 0 – No compression permitted;</li><li>• Data.HC = 1 – VJ Compression negotiated;</li><li>• Data.HC = 2,3 – Reserved.</li></ul> |

A header compression scheme can only be used when it is supported by both Service Nodes. The responder may only set Data.HC to 0 or the same value as that received from the initiator. When the same value is used, it indicates that the requested compression scheme has been negotiated and will be used for the connection.

4370 Setting Data.HC to 0 allows the responder to deny the request for that header compression scheme or force  
4371 the use of no header compression.

## 4372 **5.4.8 Service Access Point**

### 4373 **5.4.8.1 General**

4374 This section defines the service access point used by the IPv4 layer to communicate with the IPV4 SSCS.

### 4375 **5.4.8.2 Opening and closing the IPv4 SSCS**

#### 4376 **5.4.8.2.1 General**

4377 The following primitives are used to open and close the IPv4 SSCS. The IPv4 SSCS may be opened once only.  
4378 The IPv4 layer may close the IPv4 SSCS when the IPv4 interface is brought down. The IPv4 SSCS will also close  
4379 the IPv4 SSCS when the underlying MAC connection to the Base Node has been lost.

#### 4380 **5.4.8.2.2 CL\_IPv4\_ESTABLISH.request**

4381 The CL\_IPv4\_ESTABLISH.request primitive is passed from the IPv4 layer to the IPV4 SSCS. It is used when the  
4382 IPv4 layer brings the interface up.

4383 The semantics of this primitive are as follows:

4384 *CL\_IPv4\_ESTABLISH.request{AE}*

4385 The AE parameter indicates whether the interface will be authenticated and encrypted or not.

4386 On receiving this primitive, the IPV4 SSCS will form the address resolution connection to the Base Node and  
4387 join the broadcast group used for receiving/transmitting broadcast packets.

#### 4388 **5.4.8.2.3 CL\_IPv4\_ESTABLISH.confirm**

4389 The CL\_IPv4\_ESTABLISH.confirm primitive is passed from the IPV4 SSCS to the IPv4 layer. It is used to indicate  
4390 that the IPV4 SSCS is ready to access IPv4 packets to be sent to peers.

4391 The semantics of this primitive are as follows:

4392 *CL\_IPv4\_ESTABLISH.confirm{AE}*

4393 The AE parameter indicates whether the interface will be authenticated and encrypted or not.

4394 Once the IPv4 SSCS has established all the necessary connections and is ready to transmit and receive IPv4  
4395 packets, this primitive is passed to the IPv4 layer. If the IPV4 SSCS encounters an error while opening, it  
4396 responds with a CL\_IPv4\_RELEASE.confirm primitive, rather than a CL\_IPv4\_ESTABLISH.confirm.

#### 4397 **5.4.8.2.4 CL\_IPv4\_RELEASE.request**

4398 The CL\_IPv4\_RELEASE.request primitive is used by the IPv4 layer when the interface is put down. The IPV4  
4399 SSCS closes all connections so that no more IPv4 packets are received and all resources are released.

4400 The semantics of this primitive are as follows:

4401 *CL\_IPv4\_RELEASE.request{}*

4402 Once the IPV4 SSCS has released all its connections and resources it returns a CL\_IPv4\_RELEASE.confirm.

#### 4403 **5.4.8.2.5 CL\_IPv4\_RELEASE.confirm**

4404 The CL\_IPv4\_RELEASE.confirm primitive is used by the IPV4 SSCS to indicate to the IPV4 layer that the IPV4  
4405 SSCS has been closed. This can be as a result of a CL\_IPv4\_RELEASE.request primitive, a  
4406 CL\_IPv4\_ESTABLISH.request primitive, or because the MAC layer indicates the address resolution connection  
4407 has been lost, or the Service Node itself is no longer registered.

4408 The semantics of this primitive are as follows:

4409 *CL\_IPv4\_RELEASE.confirm{result}*

4410 The result parameter has the meanings defined in Table 153.

### 4411 **5.4.8.3 Unicast address management**

#### 4412 **5.4.8.3.1 General**

4413 The primitives defined here are used for address management, i.e. the registration and Unregistration of IPV4  
4414 addresses associated with this IPV4 SSCS .

4415 When there are no IPV4 addresses associated with the IPV4 SSCS, the IPV4 SSCS will only send and receive  
4416 broadcast and multicast packets; unicast packets may not be sent. However, this is sufficient for  
4417 BOOTP/DHCP operation to allow the device to gain an IPV4 address. Once an IPV4 address has been  
4418 registered, the IPV4 layer can transmit unicast packets that have a source address equal to one of its  
4419 registered addresses.

#### 4420 **5.4.8.3.2 CL\_IPv4\_REGISTER.request**

4421 This primitive is passed from the IPV4 layer to the IPV4 SSCS to register an IPV4 address.

4422 The semantics of this primitive are as follows:

4423 *CL\_IPv4\_REGISTER.request{IPv4, netmask, gateway}*

4424 The IPV4 address is the address to be registered.

4425 The netmask is the network mask, used to mask the network number from the address. The netmask is used  
4426 by the IPV4 SSCS to determine whether the packet should be delivered directly or the gateway should be  
4427 used.

4428 The gateway is an IPV4 address of the gateway to be used for packets with the IPV4 local address but the  
4429 destination address is not in the same Subnetwork as the local address.

4430 Once the IPV4 address has been registered to the Base Node, a CL\_IPv4\_REGISTER.confirm primitive is used.  
4431 If the registration fails, the CL\_IPv4\_RELEASE.confirm primitive will be used.

#### 4432 **5.4.8.3.3 CL\_IPv4\_REGISTER.confirm**

4433 This primitive is passed from the IPV4 SSCS to the IPV4 layer to indicate that a registration has been successful.

4434 The semantics of this primitive are as follows:

4435 *CL\_IPv4\_REGISTER.confirm{IPv4}*

4436 The IPv4 address is the address that was registered.

4437 Once registration has been completed, the IPv4 layer may send IPv4 packets using this source address.

4438 **5.4.8.3.4 CL\_IPv4\_UNREGISTER.request**

4439 This primitive is passed from the IPv4 layer to the IPv4 SCS to unregister an IPv4 address.

4440 The semantics of this primitive are as follows:

4441 *CL\_IPv4\_UNREGISTER.request{IPv4}*

4442 The IPv4 address is the address to be unregistered.

4443 Once the IPv4 address has been unregistered to the Base Node, a CL\_IPv4\_UNREGISTER.confirm primitive is  
4444 used. If the unregistration fails, the CL\_IPv4\_RELEASE.confirm primitive will be used.

4445 **5.4.8.3.5 CL\_IPv4\_UNREGISTER.confirm**

4446 This primitive is passed from the IPv4 SCS to the IPv4 layer to indicate that an Unregistration has been  
4447 successful.

4448 The semantics of this primitive are as follows:

4449 *CL\_IPv4\_UNREGISTER.confirm{IPv4}*

4450 The IPv4 address is the address that was unregistered.

4451 Once Unregistration has been completed, the IPv4 layer may not send IPv4 packets using this source address.

4452 **5.4.8.4 Multicast group management**

4453 **5.4.8.4.1 General**

4454 This section describes the primitives used to manage multicast groups.

4455 **5.4.8.4.2 CL\_IPv4\_IGMP\_JOIN.request**

4456 This primitive is passed from the IPv4 layer to the IPv4 SCS. It contains an IPv4 multicast address that is to  
4457 be joined.

4458 The semantics of this primitive are as follows:

4459 *CL\_IPv4\_IGMP\_JOIN.request{IPv4, AE }*

4460 The IPv4 address is the IPv4 multicast group that is to be joined.

4461 The AE parameter indicates whether messages in this group will be authenticated and encrypted or not.

4462 When the IPv4 SSCS receives this primitive, it will arrange for IPv4 packets sent to this group to be multicast  
4463 in the PRIME network and receive packets using this address to be passed to the IPv4 stack. If the IPv4 SSCS  
4464 cannot join the group, it uses the CL\_IPv4\_IGMP\_LEAVE.confirm primitive. Otherwise the  
4465 CL\_IPv4\_IGMP\_JOIN.confirm primitive is used to indicate success.

#### 4466 **5.4.8.4.3 CL\_IPv4\_IGMP\_JOIN.confirm**

4467 This primitive is passed from the IPv4 SSCS to the IPv4. It contains a result status and an IPv4 multicast address  
4468 that was joined.

4469 The semantics of this primitive are as follows:

4470 *CL\_IPv4\_IGMP\_JOIN.confirm{IPv4, AE }*

4471 The IPv4 address is the IPv4 multicast group that was joined. The IPv4 SSCS will start forwarding IPv4 multicast  
4472 packets for the given multicast group.

4473 The AE parameter indicates whether messages in this group will be authenticated and encrypted or not.

#### 4474 **5.4.8.4.4 CL\_IPv4\_IGMP\_LEAVE.request**

4475 This primitive is passed from the IPv4 layer to the IPv4 SSCS. It contains an IPv4 multicast address to be left.

4476 The semantics of this primitive are as follows:

4477 *CL\_IPv4\_IGMP\_LEAVE.request{IPv4}*

4478 The IPv4 address is the IPv4 multicast group to be left. The IPv4 SSCS will stop forwarding IPv4 multicast  
4479 packets for this group and may leave the PRIME MAC multicast group.

#### 4480 **5.4.8.4.5 CL\_IPv4\_IGMP\_LEAVE.confirm**

4481 This primitive is passed from the IPv4 SSCS to the IPv4. It contains a result status and an IPv4 multicast address  
4482 that was left.

4483 The semantics of this primitive are as follows:

4484 *CL\_IPv4\_IGMP\_LEAVE.confirm{IPv4, Result}*

4485 The IPv4 address is the IPv4 multicast group that was left. The IPv4 SSCS will stop forwarding IPv4 multicast  
4486 packets for the given multicast group.

4487 The Result takes a value from Table 153.

4488 This primitive can be used by the IPv4 SSCS as a result of a CL\_IPv4\_IGMP\_JOIN.request,  
4489 CL\_IPv4\_IGMP\_LEAVE.request or because of an error condition resulting in the loss of the PRIME MAC  
4490 multicast connection.

### 4491 **5.4.8.5 Data transfer**

#### 4492 **5.4.8.5.1 General**

4493 The following primitives are used to send and receive IPv4 packets.

**4494 5.4.8.5.2 CL\_IPv4\_DATA.request**

4495 This primitive is passed from the IPv4 layer to the IPv4 SSCS. It contains one IPv4 packet to be sent.

4496 The semantics of this primitive are as follows:

4497 *CL\_IPv4\_DATA.request{IPv4\_PDU}*

4498 The IPv4\_PDU is the IPv4 packet to be sent.

**4499 5.4.8.5.3 CL\_IPv4\_DATA.confirm**

4500 This primitive is passed from the IPv4 SSCS to the IPv4 layer. It contains a status indication and an IPv4 packet  
4501 that has just been sent.

4502 The semantics of this primitive are as follows:

4503 *CL\_IPv4\_DATA.confirm{IPv4\_PDU, Result}*

4504 The IPv4\_PDU is the IPv4 packet that was to be sent.

4505 The Result value indicates whether the packet was sent or an error occurred. It takes a value from Table 153.

**4506 5.4.8.5.4 CL\_IPv4\_DATA.indicate**

4507 This primitive is passed from the IPv4 SSCS to the IPv4 layer. It contains an IPv4 packet that has just been  
4508 received.

4509 The semantics of this primitive are as follows:

4510 *CL\_IPv4\_DATA.indicate{IPv4\_PDU }*

4511 The IPv4\_PDU is the IPv4 packet that was received.

**4512 5.5 IEC 61334-4-32 Service-Specific Convergence Sublayer (IEC**  
**4513 61334-4-32 SSCS)****4514 5.5.1 General**

4515 For all the service required, the IEC 61334-4-32 SSCS supports the DL\_DATA primitives as defined in the IEC  
4516 61334-4-32 standard. IEC 61334-4-32 should be read at the same time as this section, which is not standalone  
4517 text.

**4518 5.5.2 Overview**

4519 The IEC 61334-4-32 SSCS provides convergence functions for applications that use IEC 61334-4-32 services.  
4520 Implementations conforming to this SSCS shall offer all LLC basic and management services as specified in  
4521 IEC 61334-4-32 (1996-09 Edition), subsections 2.2.1 and 2.2.3. Additionally, the IEC 61334-4-32 SSCS specified  
4522 in this section provides extra services that help mapping this connection-less IEC 61334-4-32 LLC protocol to  
4523 the connection-oriented nature of MAC.

- A Service Node can only exchange data with the Base Node and not with other Service Nodes. This meets all the requirements of IEC 61334-4-32, which has similar restrictions.
- Each IEC 61334-4-32 SSCS session establishes a dedicated PRIME MAC connection for exchanging unicast data with the Base Node.
- The Service Node SSCS session is responsible for initiating this connection to the Base Node. The Base Node SSCS cannot initiate a connection to a Service Node.
- Each IEC 61334-4-32 SSCS listens to a PRIME broadcast MAC connection dedicated to the transfer of IEC 61334-4-32 broadcast data from the Base Node to the Service Nodes. This broadcast connection is used when applications in the Base Node using IEC 61334-4-32 services make a transmission request with the Destination\_address used for broadcast or the broadcast SAP functions are used. When there are multiple SSCS sessions within a Service Node, one PRIME broadcast MAC connection is shared by all the SSCS sessions.
- A CPCS session is always present with a IEC 61334-4-32 SSCS session. The SPCS sublayer functionality is as specified in Section 5.2.2. Thus, the MSDUs generated by IEC 61334-4-32 SSCS are always less than *macSARSize* bytes and application messages shall not be longer than *CIMaxAppPktSize*.

### 5.5.3 Address allocation and connection establishment

Each 4-32 connection will be identified with the "Application unique identifier" that will be communicating through this 4-32 connection. It is the scope of the communication profile based on these lower layers to define the nature and rules for, this unique identifier. Please refer to the future prTS/EN52056-8-4 for the DLMS/COSEM profile unique identifier. As long as the specification of the 4-32 Convergence layer concerns this identifier will be called the "Device Identifier".

The protocol stack as defined in IEC 61334 defines a Destination address to identify each device in the network. This Destination address is specified beyond the scope of the IEC 61334-4-32 document. However, it is used by the document. So that PRIME devices can make use of the 4-32 layer, this Destination address is also required and is specified here. For more information about this Destination address, please see IEC 61334-4-1 section 4.3, MAC Addresses.

The Destination address has a scope of one PRIME Subnetwork. The Base Node 4-32 SSCP layer is responsible for allocating these addresses dynamically and associating the Device Identifier of the Service Nodes SSCP session device with the allocated Destination address, according to the IEC-61334-4-1 standard. The procedure is as follows:

When the Service Node IEC 61334-4-32 SSCS session is opened by the application layer, it passes the Device Identifier of the device. The IEC 61334-4-32 SSCS session then establishes its unicast connection to the Base Node. This unicast connection uses the PRIME MAC TYPE value TYPE\_CL\_432, as defined in Table 151. The connection request packet sent from the Service Node to the Base Node contains a data parameter. This data parameter contains the Device Identifier. The format of this data is specified in section 5.5.4.2.

On receiving this connection request at the Base Node, the Base Node allocates a unique Subnetwork Destination address to the Service Nodes SSCS session. The Base Node sends back a PRIME MAC connection response packet that contains a data parameter. This data parameter contains the allocated Destination address and the address being used by the Base Node itself. The format of this data parameter is defined in

4564 section 5.5.4.2. A 4-32 CL SAP primitive is used in the Base Node to indicate this new Service Node SSCS  
4565 session mapping of Device Identifier and Destination\_address to the 4-32 application running in the Base  
4566 Node.

4567 On receiving the connection establishment and the Destination\_address passed in the PRIME MAC  
4568 connection establishment packet, the 4-32 SSCS session confirms to the application that the Convergence  
4569 layer session has been opened and indicates the Destination\_address allocated to the Service Node SSCS  
4570 session and the address of the Base Node. The Service Node also opens a PRIME MAC broadcast connection  
4571 with LCID equal to LCI\_CL\_432\_BROADCAST, as defined in Table 152, if no other SSCS session has already  
4572 opened such a broadcast connection. This connection is used to receive broadcast packets sent by the Base  
4573 Node 4-32 Convergence layer to all Service Node 4-32 Convergence layer sessions.

4574 If the Base Node has allocated all its available Destination\_addresses, due to the exhaustion of the address  
4575 space or implementation limits, it should simply reject the connection request from the Service Node. The  
4576 Service Node may try to establish the connection again. However, to avoid overloading the PRIME  
4577 Subnetwork with such requests, it should limit such connection establishments to one attempt per minute  
4578 when the Base Node rejects a connection establishment.

4579 When the unicast connection between a Service Node and the Base Node is closed (e.g. because the  
4580 Convergence layer on the Service Node is closed or the PRIME MAC level connection between the Service  
4581 Node and the Base Node is lost), the Base Node will deallocate the Destination\_address allocated to the  
4582 Service Node SSCS session. The Base Node will use a 4-32 CL SAP (CL\_432\_Leave.indication) primitive to  
4583 indicate the deallocation of the Destination\_address to the 4-32 application running on the Base Node

## 4584 **5.5.4 Connection establishment data format**

### 4585 **5.5.4.1 General**

4586 As described in section 5.5.3, the MAC PRIME connection data is used to transfer the Device Identifier to the  
4587 Base Node and the allocated Destination\_address to the Service Node SSCS session. This section describes  
4588 the format used for this data.

### 4589 **5.5.4.2 Service Node to Base Node**

4590 The Service Node session passes the Device Identifier to the Base Node as part of the connection  
4591 establishment request. The format of this message is shown in Table 85.

4592 **Table 85 - Connection Data sent by the Service Node**

| Name    | Length   | Description                                                                                                                                                                                             |
|---------|----------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Data.SN | n-Octets | Device Identifier.<br>"COSEM logical device name" of the "Management logical device" of the DLMS/COSEM device as specified in the DLMS/COSEM, which will be communicating through this 4-32 connection. |



### 5.5.4.3 Base Node to Service Node

The Base Node passes the allocated Destination\_address to the Service Node session as part of the connection establishment request. It also gives its own address to the Service Node. The format of this message is shown in Table 86.

Table 86 - Connection Data sent by the Base Node

| Name            | Length  | Description                                                               |
|-----------------|---------|---------------------------------------------------------------------------|
| <i>Reserved</i> | 4-bits  | Reserved. Should be encoded as zero in this version of the specification. |
| Data.DA         | 12-bits | Destination_address allocated to the Service Node.                        |
| <i>Reserved</i> | 4-bits  | Reserved. Should be encoded as zero in this version of the specification. |
| Data.BA         | 12-bits | Base_address used by the Base Node.                                       |

## 5.5.5 Packet format

The packet formats are used as defined in IEC 61334-4-32, Clause 4, LLC Protocol Data Unit Structure (LLC\_PDU).

## 5.5.6 Service Access Point

### 5.5.6.1 Opening and closing the Convergence layer at the Service Node

#### 5.5.6.1.1 CL\_432\_ESTABLISH.request

This primitive is passed from the application to the 4-32 Convergence layer. It is used to open a Convergence layer session and initiate the process of registering the Device Identifier with the Base Node and the Base Node allocating a Destination\_address to the Service Node session.

The semantics of this primitive are as follows:

*CL\_432\_ESTABLISH.request{ DeviceIdentifier, AE }*

The Device Identifier is that of the device to be registered with the Base Node.

The AE parameter indicates whether the session will be authenticated and encrypted or not.

If the Device Identifier is registered and the Convergence layer session is successfully opened, the primitive CL\_432\_ESTABLISH.confirm is used. If an error occurs the primitive CL\_432\_RELEASE.confirm is used.

#### 5.5.6.1.2 CL\_432\_ESTABLISH.confirm

This primitive is passed from the 4-32 Convergence layer to the application. It is used to confirm the successful opening of the Convergence layer session and that data may now be passed over the Convergence layer.

The semantics of this primitive are as follows:

4619 *CL\_432\_ESTABLISH.confirm{ DeviceIdentifier, Destination\_address, Base\_address, AE }*

4620 The Device Identifier is used to identify which CL\_432\_ESTABLISH.request this CL\_432\_ESTABLISH.confirm is  
4621 for.

4622 The Destination\_address is the address allocated to the Service Node 4-32 session by the Base Node.

4623 The Base\_address is the address being used by the Base Node.

4624 The AE parameter indicates whether the session will be authenticated and encrypted or not.

#### 4625 **5.5.6.1.3 CL\_432\_RELEASE.request**

4626 This primitive is passed from the application to the 4-32 Convergence layer. It is used to close the  
4627 Convergence layer and release any resources it may be holding.

4628 The semantics of this primitive are as follows:

4629 *CL\_432\_RELEASE.request{Destination\_address}*

4630 The Destination\_address is the address allocated to the Service Node 4-32 session which is to be closed.

4631 The Convergence layer will use the primitive CL\_432\_RELEASE.confirm when the Convergence layer session  
4632 has been closed.

#### 4633 **5.5.6.1.4 CL\_432\_RELEASE.confirm**

4634 This primitive is passed from the 4-32 Convergence layer to the application. The primitive tells the application  
4635 that the Convergence layer session has been closed. This could be because of a local or remote  
4636 CL\_432\_RELEASE.request, or because an error has occurred, forcing the closure of the Convergence layer  
4637 session.

4638 The semantics of this primitive are as follows:

4639 *CL\_432\_RELEASE.confirm{Destination\_address, result}*

4640 The Handle identifies the session which has been closed.

4641 The result parameter has the meanings defined in Table 153.

#### 4642 **5.5.6.2 Opening and closing the Convergence layer at the Base Node**

4643 No service access point primitives are defined at the Base Node for opening or closing the local Convergence  
4644 layer. None are required since the 4-32 application in the Base Node does not need to pass any information  
4645 to the 4-32 Convergence layer in the Base Node.

4646 Note: The remote closure of the Converge layer in Service Node it is an optional functionality that can be  
4647 triggered by the Base Node through the same primitives described in sections 5.5.6.1.3 and 5.5.6.1.4 for the  
4648 Service Node.

### 4649 5.5.6.3 Base Node indications

#### 4650 5.5.6.3.1 General

4651 The following primitives are used in the Base Node 4-32 Convergence layer to indicate events to the 4-32  
4652 application in the Base Node. They indicate when a Service Node session has joined or left the network.

#### 4653 5.5.6.3.2 CL\_432\_JOIN.indicate

4654 *CL\_432\_JOIN.indicate{ Device Identifier, Destination\_address, AE}*

4655 The Device Identifier is that of the device connected to the Service Node that has just joined the network.

4656 The Destination\_address is the address allocated to the Service Node by the Base Node.

4657 The AE parameter indicates whether messages in this session will be authenticated and encrypted or not.

#### 4658 5.5.6.3.3 CL\_432\_LEAVE.indicate

4659 *CL\_432\_LEAVE.indicate{Destination\_address}*

4660 The Destination\_address is the address of the Service Node session that just left the network.

### 4661 5.5.6.4 Data Transfer Primitives

4662 The data transfer primitives are used as defined in IEC 61334-4-32, sections 2.2, 2.3, 2.4 and 2.11, LLC Service  
4663 Specification. As stated earlier, PRIME 432 SCSs make the use of IEC61334-4-32 DL\_Data service (.req, .conf,  
4664 .ind) for carrying out all the data involved during data transfer. Only DL\_DATA service is mandatory

4665

## 4666 5.6 IPv6 Service-Specific Convergence Sublayer (IPv6 SCS)

### 4667 5.6.1 Overview

#### 4668 5.6.1.1 General

4669 The IPv6 convergence layer provides an efficient method for transferring IPv6 packets over the PRIME  
4670 network.

4671 A Service Node can pass IPv6 packets to the Base Node or directly to other Service Nodes.

4672 By default, the Base Node acts as a router between the PRIME subnet and the backbone network. All the  
4673 Base Nodes must have at least this connectivity capability. Any other node inside the Subnetwork can also  
4674 act as a gateway. The Base Node could also act as a NAT router. However given the abundance of IPv6  
4675 addresses this is not expected. How the Base Node connects to the backbone is beyond the scope of this  
4676 standard.

#### 4677 5.6.1.2 IPv6 unicast addressing assignment

- 4678 • IPv6 Service Nodes (and Base Nodes) shall support the standard IPv6 protocol, as described in  
4679 RFC 2460.

- IPv6 Service Nodes (and Base Nodes) shall support the standard IPv6 addressing architecture, as described in RFC 4291.
- IPv6 Service Nodes (and Base Nodes) shall support global unicast IPv6 addresses, link-local IPv6 addresses and multicast IPv6 addresses, as described in RFC 4291.
- IPv6 Service Nodes (and Base Nodes) shall support automatic address configuration using stateless address configuration [RFC 2462]. They may also support automatic address configuration using stateful address configuration [RFC 3315] and they may support manual configuration of IPv6 addresses. The decision of which address configuration scheme to use is deployment specific.
- Service Node shall support DHCPv6 client, when Base Nodes have to support DHCPv6 server as described in RFC 3315 for stateless address configuration

### 5.6.1.3 Address management in PRIME Subnetwork

Packets are routed in PRIME Subnetwork according to the node identifier NID. Node identifier is a combination of Service Node's LNID and SID (see section 4.2). The Base Node is responsible of assigning LNID to Service Nodes. During the registration process which leads to a LNID assignment to the related Service Node, the Base Node registers the Service Node EUI-48, and the assigned LNID together with SID.

At the convergence layer level, addressing is performed using the EUI-48 of the related Service Node. The role of the convergence sublayer is to resolve the IPv6 address into EUI-48 of the Service Node. This is done using the address resolution service set of the Base Node.

### 5.6.1.4 Role of the Base Node

At the convergence sublayer level, the Base Node maintains a table containing all the IPv6 unicast addresses and the EUI-48 related to them. One of the roles of the Base Node is to perform IPv6 to EUI-48 address resolution. Each Service Node belonging to the Subnetwork managed by the Base Node, registers its IPv6 address and EUI-48 address with the Base Node. Other Service Nodes can then query the Base Node to resolve an IPv6 address into a EUI-48 address. This requires the establishment of a dedicated connection to the Base Node for address resolution, which is shared by both IPv4 and IPv6 address resolution.

Optionally UDP/IPv6 headers may be compressed. Currently one header compression technique is described in the present specification that used for transmission of IPv6 packets over IEEE 802.15.4 networks, as defined in RFC6282. This is also known as LOWPAN\_IPHC1.

The multicasting of IPv6 packets is supported using the MAC multicast mechanism

## 5.6.2 IPv6 Convergence layer

### 5.6.2.1 Overview

#### 5.6.2.1.1 General

The convergence layer has a number of connection types. For address resolution there is a connection to the Base Node. For IPv6 data transfer there are one up to many connections per destination node, depending on how many source's and destination's IPv6 addresses are in use. This is shown in Figure 137.

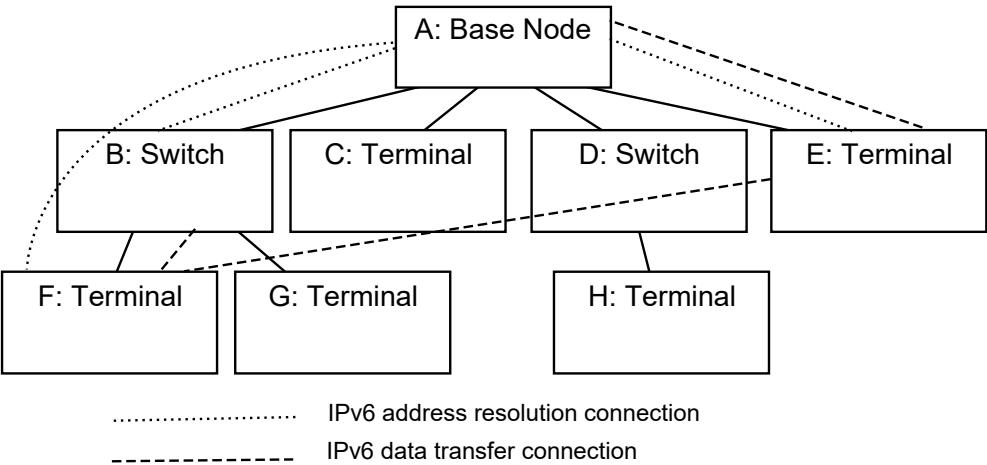


Figure 137 - IPv6 SCS connection example

Here, nodes B, E and F have address resolution connections to the Base Node. Node E has a data connection to the Base Node and node F. Node F is also has a data connection to node B. The figure does not show broadcast-traffic and multicast-traffic connections.

5.6.2.1.2 Routing in the Subnetwork

Routing IPv6 packets is the scope of the Convergence layer. In other words, the convergence layer will decide whether the packet should be sent directly to another Service Node or forwarded to the configured gateway depending on the IPv6 destination address.

Although IPv6 is a connectionless protocol, the IPv6 convergence layer is connection-oriented. Once address resolution has been performed, a connection is established between the source and destination Service Nodes for the transfer of IP packets. This connection is maintained all the time the traffic is being transferred and may be removed after a period of inactivity.

5.6.2.1.3 SAR

The CPCS sublayer shall always be present with the IPv6 convergence layer allowing segmentation and reassembly facilities. The SAR sublayer functionality is given in Section 5.2. Thus, the MSDUs generated by the IPv6 convergence layer are always less than macSARSize bytes and application messages are expected to be no longer than CIMaxAppPktSize.

5.6.3 IPv6 Address Configuration

5.6.3.1 Overview

The Service Nodes may use statically configured IPv6 addresses, link local addresses, stateless or stateful auto-configuration according to RFC 2462, or DHCPv6 to obtain IPv6 addresses. All the Nodes shall support the unicast link local address, in addition with at least one other address from those described below, and multicast addresses, if ever the node belong to multicast groups.

5.6.3.2 Interface identifier

In order to make use of stateless address auto configuration and link local addresses it is necessary to define how the Interface identifier, as defined in RFC4291, is derived. Each PRIME node has a unique EUI-48. This

4744 EUI-48 is converted into a EUI-64 in the same way as for Ethernet networks as defined in RFC2464. This EUI-  
4745 64 is then used as the Interface Identifier.

### 4746 **5.6.3.3 IPv6 Link local address configuration**

4747 The IPv6 Link local address of a PRIME interface is formed by appending the Interface Identifier as defined  
4748 above to the Prefix FE80::/64.

### 4749 **5.6.3.4 Stateless address configuration**

4750 An IPv6 address prefix used for stateless auto configuration, as defined in RFC4862, of a PRIME interface shall  
4751 have a length of 64 bits. The IPv6 prefix is obtained by the Service Nodes from the Base Node via Router  
4752 Advertisement messages (RFC 4861), which are send periodically or on request by the Base Node.

### 4753 **5.6.3.5 Stateful address configuration**

4754 An IPv6 address can be alternatively configured using DHCPv6, as described in RFC 3315. DHCPv6 can provide  
4755 a device with addresses assigned by a DHCPv6 server and other configuration information, which are carried  
4756 in options.

### 4757 **5.6.3.6 Multicast address**

4758 IPv6 Service Nodes (and Base Nodes) shall support the multicast IPv6 addressing, as described in RFC 4291  
4759 section 2.7.

### 4760 **5.6.3.7 Address resolution**

#### 4761 **5.6.3.7.1 Overview**

4762 The IPv6 layer will present the convergence layer with an IPv6 packet to be transferred. The convergence  
4763 layer is responsible for determining which Service Node the packet should be delivered to, using the IPv6  
4764 addresses in the packet. The convergence layer shall then establish a connection to the destination if one  
4765 does not already exist so that the packet can be transferred. Two classes of IPv6 addresses can be used and  
4766 the following section describes how these addresses are resolved into PRIME EUI-48 addresses. It should be  
4767 noted that IPv6 does not have a broadcast address. However broadcasting is possible using multicast all  
4768 nodes addresses.

4769

#### 4770 **5.6.3.7.2 Unicast address**

##### 4771 **5.6.3.7.2.1 General**

4772 IPv6 unicast addresses shall be resolved into PRIME unicast EUI-48 addresses. The Base Node maintains a  
4773 central database Node of IPv6 addresses and EUI-48 addresses. Address resolution functions are performed  
4774 by querying this database. The Service Node shall establish a connection to the address resolution service  
4775 running on the Base Node, using the TYPE value TYPE\_CL\_IPv6\_AR. No data should be passed in the  
4776 connection establishment. Using this connection, the Service Node can use two mechanisms as defined in  
4777 the present specification.

4778

**4779 5.6.3.7.2.2 Address registration and deregistration**

4780 A Service Node uses the AR\_REGISTERv6\_S message to register an IPv6 address and the corresponding EUI-  
4781 48 address. The Base Node will acknowledge an AR\_REGISTERv6\_B message. The Service Node may register  
4782 multiple IPv6 addresses for the same EUI-48.

4783 A Service Node uses the AR\_UNREGISTERv6\_S message to unregister an IPv6 address and the corresponding  
4784 EUI-48 address. The Base Node will acknowledge with an AR\_UNREGISTERv6\_B message.

4785 When the address resolution connection between the Service Node and the Base Node is closed, the Base  
4786 Node should remove all addresses associated with that connection.

4787

**4788 5.6.3.7.2.3 Address lookup**

4789 A Service Node uses the AR\_LOOKUPv6\_S message to perform a lookup. The message contains the IPv6  
4790 address to be resolved. The Base Node will respond with an AR\_LOOKUPv6\_B message that contains an error  
4791 code and, if there is no error, the EUI-48 associated with the IPv6 address. If the Base Node has multiple  
4792 entries in its database Node for the same IPv6 address, the possible EUI-48 returned is undefined.

4793 It should be noted that, for the link local addresses, due to the fact that the EUI-48 can be obtained from the  
4794 IPv6 address, the lookup can simply return this value by extracting it from the IPv6 address.

**4795 5.6.3.7.3 Multicast address**

4796 Multicast IPv6 addresses are mapped to connection handles (ConnHandle) by the Convergence Layer (see  
4797 Table 88).

**4798 5.6.3.7.4 Retransmission of address resolution packets**

4799 The connection between the Service Node and the Base Node for address resolution is not reliable. The MAC  
4800 ARQ is not used. The Service Node is responsible for making retransmissions if the Base Node does not  
4801 respond in one second. It is not considered an error when the Base Node receives the same registration  
4802 requests multiple times or is asked to remove a registration that does not exist. These conditions can be the  
4803 result of retransmissions.

**4804 5.6.4 IPv6 Packet Transfer****4805 5.6.4.1 Unicast transfer****4806 5.6.4.1.1 Gateway info**

4807 The two endpoints exchanging unicast IPv6 datagrams may both be placed within a PRIME network, or one  
4808 of them can be outside. In the latter case a PRIME node shall act as gateway to route the IPv6 traffic in and  
4809 out the PRIME network. It shall take 6LoWPAN encoded outgoing datagrams and reconstruct the full IPv6  
4810 counterparts before sending them to the outside; and, viceversa, it shall take incoming IPv6 datagrams and  
4811 6LoWPAN encoding them before injecting them into the PRIME network.



The IPv6 layer registers its addresses using the primitive CL\_IPv6\_Register.request (5.6.9.3.2). For each one, the subnet mask and the gateway IPv6 address are provided. As soon as the address registration (5.6.4.1.2) to the BN is successful, the SSCS must store these info in an internal table like the one in Table 87.

**Table 87 – Node addresses table entry**

| Parameter                | Description                                                          |
|--------------------------|----------------------------------------------------------------------|
| CL_IPv6_Addr.Local_IP    | Node's registered IPv6 address.                                      |
| CL_IPv6_Addr.Subnet_Mask | Subnet mask of the IPv6 subnetwork the node IPv6 address belongs to. |
| CL_IPv6_Addr.Gateway_IP  | IPv6 address of the subnet's gateway.                                |

#### 5.6.4.1.2 Both endpoints in the same PRIME network

For packets to be transferred, a connection needs to be established between the source and destination nodes. Considering that any node may be given with one to many IPv6 addresses, a different connection shall be established between two IPv6-communicating PRIME nodes for each pair of their IPv6 addresses exchanging datagrams. The IPv6 convergence layer will examine each IP packet to determine pair of source and destination IPv6 addresses. By matching the destination IPv6 address against the subnet mask stored in Table 87 for the datagram's source IPv6 address, the SSCS determines that both endpoints are two PRIME nodes in the same PRIME network:

*source IPv6 addr & CL\_IPv6\_Addr.Subnet\_Mask == destination IPv6 addr & CL\_IPv6\_Addr.Subnet\_Mask*

If a connection between PRIME nodes owning these addresses has already been established by the SSCS, the packet is simply sent over that connection. To verify this, the convergence layer keeps a table for each connection it has with information shown in Table 88.

**Table 88– IPv6 convergence layer table entry**

| Parameter             | Description                                   |
|-----------------------|-----------------------------------------------|
| CL_IPv6_Con.Local_IP  | Local IP address of this connection           |
| CL_IPv6_Con.Remote_IP | Remote IP address of this connection          |
| CL_IPv6_Con.ConHandle | MAC Connection handle for the connection      |
| CL_IPv6_Con.LastUsed  | Timestamp of last packet received/transmitted |
| CL_IPv6_Con           | Header Compression scheme being used          |

The convergence layer may close a connection when it has not been used for an implementation-defined time period. When the connection is closed the entry for the connection is removed at both ends of the connection.



When a connection to the destination does not exist, more work is necessary. The address resolution service is used to determine the EUI-48 address of the remote IP address. When the Base Node replies with the EUI-48 address of the destination Service Node, a MAC connection is established to the remote device. The TYPE value of this connection is TYPE\_CL\_IPv6\_DATA. The data passed in the request message is defined in section 5.6.8.3. Both local source IP address and the destination remote IPv6 address are provided so that the remote device can properly set up its own IPv6 convergence layer table entry, thus adding the new connection to its cache of connections for sending data in the opposite direction. Once the connection has been established, the IP packet can be sent.

#### 5.6.4.1.3 One endpoint outside the PRIME network

If the communication involves an endpoint external to the PRIME network, datagrams shall pass through a border PRIME node acting as the gateway. In this scenario a PRIME connection shall be established between the gateway and the PRIME node, and 6LoWPAN compressed datagrams shall be transmitted over it.

Following subsections cover the connection establishment and usage in the two possible scenarios, depending on which endpoint sends the first datagram.

##### 5.6.4.1.3.1 Connection initiated by the PRIME endpoint

As for the PRIME internal data exchange (5.6.4.1.2), SSCS discovers whether the destination is outside the PRIME network by matching the destination IPv6 address against the netmask associated with the source IPv6 address in Table 87

*source IPv6 addr & CL\_IPv6\_Addr.Subnet\_Mask != destination IPv6 addr & CL\_IPv6\_Addr.Subnet\_Mask*

Once the gateway IPv6 address is retrieved from the Table 87's row of the source IPv6 address, SSCS looks up the addresses couple (source IPv6 address, gateway IPv6 address) in the Table 88 to get the connection ID with the gateway – if a connection has not been established yet, same procedures (address resolution, connection establishment and entry addition in Table 88 ) as with PRIME network internal communication (5.6.4.1.2) shall be undergone, using the gateway IPv6 address as the remote endpoint address. Once the connection ID is retrieved, the 6LoWPAN encoded version of the datagram shall be sent over it.

##### 5.6.4.1.3.2 Connection initiated by the external endpoint

The SSCS discovers that the datagram is coming from the outside by failing at looking up the source IPv6 address in Table 87.

By scanning rows in Table 87, the SSCS finds the gateway IPv6 address the PRIME node owning the destination IPv6 address refers to. It is the IPv6 address stored in the table's entry for which the following relationship holds:

*CL\_IPv6\_Addr.Gateway\_IP & CL\_IPv6\_Addr.Subnet\_Mask ==  
destination IPv6 addr & CL\_IPv6\_Addr.Subnet\_Mask*

Once the gateway IPv6 address is retrieved, SSCS looks up the addresses couple (gateway IPv6 address, destination IPv6 address) in the Table 88 to get the connection ID with the destination PRIME node – if a connection has not been established yet, same procedures (address resolution, connection establishment and entry addition in Table 88) as with PRIME network internal communication (5.6.4.1.2) shall be undergone,

using the gateway IPv6 address as the local endpoint address. Once the connection ID is retrieved, the 6LoWPAN encoded version of the datagram shall be sent over it.

#### 5.6.4.2 Multicast transfer

To join a multicast group, CL uses the MAC\_JOIN.request primitive with the IPv6 address specified in the data field. A corresponding MAC\_JOIN.Confirm primitive will be generated by the MAC after completion of the join process. The MAC\_Join.Confirm primitive will contain the result (success/failure) and the corresponding ConnHandle to be used by the CL. The MAC layer will handle the transfer of data for this connection using the appropriate LCIDs. To leave the multicast group, the CL at the service node shall use the MAC\_LEAVE.Request{ConnHandle} primitive.

To send an IPv6 multicast packet, the CL will simply send the packet to the group, using the allocated ConnHandle.

Table 89 – Ipv6 convergence layer multicast table entry

| Parameter             | Description                                |
|-----------------------|--------------------------------------------|
| CL_Ipv6_Mul.Address   | Multicast Ipv6 address for this connection |
| CL_Ipv6_Mul.ConHandle | MAC Connection handle for the connection   |

#### 5.6.5 Segmentation and reassembly

The Ipv6 convergence layer should support Ipv6 packets with an MTU of at least 1500 bytes. This requires the use of the common part convergence sublayer segmentation and reassembly service.

#### 5.6.6 Compression

Any PRIME device being compliant with this Service-Specific Convergence Sublayer shall be able to decode any valid 6LoWPAN encoded packet.

All the Service Nodes and the Base Node shall support IPv6 Header Compression using source and destination Addresses stateless compression as defined in RFC 6282. Source and destination IPv6 addresses using stateful compressions shall also be supported, but the way contexts are shared is outside the scope of this document.

As far as the stateless compression of either source address or unicast destination address is concerned, the 6LoWPAN implementation in this Service-Specific Convergence Sublayer shall allow following modes only (refer to RFC 6282 for fields' meaning):

- Full address is carried inline (e.g. no compression) (SAM=0b00 and/or DAM=0b00).
- Address is fully elided (SAM=0b11 and/or DAM=0b11).

Remaining two modes – address compressed down to 64 bits (SAM=0b01 and/or DAM=0b01) and down to 16 bits (SAM=0b10 and/or DAM=0b10) – are forbidden as PRIME own characteristics don't allow for such kind of compression. The full address compression (SAM=0b11 and/or DAM=0b11) is enabled by the information added in the Table 89 once the connection between the two is established. Such a table enables

a direct mapping between the connection handle and the IPv6 addresses. When a node receives a PRIME packet, it uses its connection handle as table's lookup key: once found, it can retrieve both IPv6 source and destination addresses and hence restore them in the IPv6 datagram. If both IPv6 endpoints are in the same PRIME network, both their addresses may be fully compressed by the sender. When instead one node is outside the PRIME network, and hence the PRIME connection involves the gateway, the outer node's IPv6 address cannot be compressed, because such an address is not the one of the PRIME node being involved in the connection; in such a situation, only the IPv6 address of the PRIME endpoint can be fully elided.

### 5.6.7 Quality of Service Mapping

The PRIME MAC specifies that the contention-based access mechanism supports 4 priority levels (1-4). Level 1 is used for MAC control messages, but not exclusively so.

IPv6 packets include a Traffic Class field in the header to indicate the QoS the packet would like to receive. This traffic class can be used in the same way that IPv4 TOS (see [7]). That is, three bits of the TOS indicate the IP Precedence. The following table specifies how the IP Precedence is mapped into the PRIME MAC priority.

Table 90 – Mapping Ipv6 precedence to PRIME MAC priority

| IP Precedence              | MAC Priority |
|----------------------------|--------------|
| 000 – Routine              | 3            |
| 001 – Priority             | 3            |
| 010 – Immediate            | 2            |
| 011 – Flash                | 2            |
| 100 – Flash Override       | 1            |
| 101 – Critical             | 1            |
| 110 – Internetwork Control | 0            |
| 111 – Network Control      | 0            |

### 5.6.8 Packet formats and connection data

#### 5.6.8.1 Overview

This section defines the format of convergence layer PDUs.

#### 5.6.8.2 Address resolution PDU

##### 5.6.8.2.1 General

The following PDUs are transferred over the address resolution connection between the Service Node and the Base Node. The following sections define a number of AR.MSG values. All other values are reserved for later versions of this standard.

**5.6.8.2.2 AR\_REGISTERv6\_S**

Table 91 shows the address resolution register message sent from the Service Node to the Base Node.

Table 91 - AR\_REGISTERv6\_S message format

| Name      | Length   | Description                                                                                              |
|-----------|----------|----------------------------------------------------------------------------------------------------------|
| AR.MSG    | 8-bits   | Address Resolution Message Type <ul style="list-style-type: none"><li>For AR_REGISTERv6_S = 16</li></ul> |
| AR.IPv6   | 128-bits | IPv6 address to be registered                                                                            |
| AR.EUI-48 | 48-bits  | EUI-48 to be registered                                                                                  |

**5.6.8.2.3 AR\_REGISTERv6\_B**

Table 92 shows the address resolution register acknowledgment message sent from the Base Node to the Service Node.

Table 92 - AR\_REGISTERv6\_B message format

| Name      | Length   | Description                                                                                              |
|-----------|----------|----------------------------------------------------------------------------------------------------------|
| AR.MSG    | 8-bits   | Address Resolution Message Type <ul style="list-style-type: none"><li>For AR_REGISTERv6_B = 17</li></ul> |
| AR.IPv6   | 128-bits | IPv6 address registered                                                                                  |
| AR.EUI-48 | 48-bits  | EUI-48 registered                                                                                        |

The AR.IPv6 and AR.EUI-48 fields are included in the AR\_REGISTERv6\_B message so that the Service Node can perform multiple overlapping registrations.

**5.6.8.2.4 AR\_UNREGISTERv6\_S**

Table 93 shows the address resolution unregister message sent from the Service Node to the Base Node.

Table 93 - AR\_UNREGISTERv6\_S message format

| Name    | Length   | Description                                                                                                |
|---------|----------|------------------------------------------------------------------------------------------------------------|
| AR.MSG  | 8-bits   | Address Resolution Message Type <ul style="list-style-type: none"><li>For AR_UNREGISTERv6_S = 18</li></ul> |
| AR.IPv6 | 128-bits | IPv6 address to be unregistered                                                                            |

|           |         |                           |
|-----------|---------|---------------------------|
| AR.EUI-48 | 48-bits | EUI-48 to be unregistered |
|-----------|---------|---------------------------|

#### 5.6.8.2.5 AR\_UNREGISTERv6\_B

Table 94 shows the address resolution unregister acknowledgment message sent from the Base Node to the Service Node.

Table 94 - AR\_UNREGISTERv6\_B message format

| Name      | Length   | Description                                                                                                |
|-----------|----------|------------------------------------------------------------------------------------------------------------|
| AR.MSG    | 8-bits   | Address Resolution Message Type <ul style="list-style-type: none"><li>For AR_UNREGISTERv6_B = 19</li></ul> |
| AR.IPv6   | 128-bits | IPv6 address unregistered                                                                                  |
| AR.EUI-48 | 48-bits  | EUI-48 unregistered                                                                                        |

The AR.IPv6 and AR.EUI-48 fields are included in the AR\_UNREGISTERv6\_B message so that the Service Node can perform multiple overlapping unregistrations.

#### 5.6.8.2.6 AR\_LOOKUPv6\_S

Table 95 shows the address resolution lookup message sent from the Service Node to the Base Node.

Table 95 - AR\_LOOKUPv6\_S message format

| Name    | Length   | Description                                                                                            |
|---------|----------|--------------------------------------------------------------------------------------------------------|
| AR.MSG  | 8-bits   | Address Resolution Message Type <ul style="list-style-type: none"><li>For AR_LOOKUPv6_S = 20</li></ul> |
| AR.IPv6 | 128-bits | IPv6 address to lookup                                                                                 |

#### 5.6.8.2.7 AR\_LOOKUPv6\_B

Table 96 shows the address resolution lookup response message sent from the Base Node to the Service Node.

Table 96 - AR\_LOOKUPv6\_B message format

| Name   | Length | Description                     |
|--------|--------|---------------------------------|
| AR.MSG | 8-bits | Address Resolution Message Type |

|           |          |                                                                                                                                                                                                  |
|-----------|----------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|           |          | <ul style="list-style-type: none"> <li>For AR_LOOKUPv6_B = 21</li> </ul>                                                                                                                         |
| AR.IPv6   | 128-bits | IPv6 address looked up                                                                                                                                                                           |
| AR.EUI-48 | 48-bits  | EUI-48 for IPv6 address                                                                                                                                                                          |
| AR.Status | 8-bits   | Lookup status, indicating if the address was found or an error occurred. <ul style="list-style-type: none"> <li>0 = found, AR.EUI-48 valid.</li> <li>1 = unknown, AR.EUI-48 undefined</li> </ul> |

4960

4961 The lookup may fail if the requested address has not been registered. In that case, AR.Status will have a value  
 4962 equal to 1, and the contents of AR.EUI-48 will be undefined. The lookup is only successful when AR.Status is  
 4963 zero. In that case, the EUI-48 field contains the resolved address.

### 4964 5.6.8.3 IPv6 Packet format

#### 4965 5.6.8.3.1 General

4966 The following PDU formats are used for transferring IPv6 packets between Service Nodes.

4967

#### 4968 5.6.8.3.2 No header compression

4969 When no header compression is used, the IP packet is simply sent as it is, without any header.

4970 Table 97 - IPv6 Packet format without header compression

| Name     | Length   | Description     |
|----------|----------|-----------------|
| IPv6.PKT | n-octets | The IPv6 Packet |

#### 4971 5.6.8.3.3 Header compression

4972 When LOWPAN\_IPHC1 header compression is used, the UDP/IPv6 packet is sent as shown in Table 98.

4973 Table 98 - UDP/IPv6 Packet format with LOWPAN\_IPHC1 header compression and LOWPAN\_NHC

| Name        | Length     | Description                                                                                                   |
|-------------|------------|---------------------------------------------------------------------------------------------------------------|
| IPv6.IPHC   | 2-octet    | Dispatch + LOWPAN_IPHC encoding. With bit 5=1 indicating that the next is compressed ,using LOWPAN_NHC format |
| IPv6.ncIPv6 | n.m-octets | Non-Compressed IPv6 fields (or elided)                                                                        |
| IPv6.HC_UDP | 1-octet    | Next header encoding                                                                                          |

|                  |            |                           |
|------------------|------------|---------------------------|
| IPv6.ncUDP       | n.m-octets | Non-Compressed UDP fields |
| <i>Padding</i>   | 0.m-octets | Padding to byte boundary  |
| <i>IPv6.DATA</i> | n-octets   | UDP data                  |

4974

4975 Note that these fields are not necessarily aligned to byte boundaries. For example the IPv6.ncIPv6 field can  
4976 be any number of bits. The IPv6.IPHC\_UDP field follows directly afterwards, without any padding. Padding is  
4977 only applied at the end of the complete compressed UDP/IPv6 header such that the UDP data is byte aligned.

4978 When the IPv6 packet contains data other than UDP the following packet format is used as shown in Table  
4979 99.

4980

**Table 99 - IPv6 Packet format with LOWPAN\_IPHC negotiated header compression**

| Name             | Length     | Description                                                                      |
|------------------|------------|----------------------------------------------------------------------------------|
| IPv6.IPHC        | 2-octet    | HC encoding. Bits 5 contain 0 indicating the next header byte is not compressed. |
| IPv6.ncIPv6      | n.m-octets | Non-Compressed IPv6 fields                                                       |
| <i>Padding</i>   | 0.m-octets | Padding to byte boundary                                                         |
| <i>IPv6.DATA</i> | n-octets   | IP Data                                                                          |

#### 4981 5.6.8.4 Connection data

##### 4982 5.6.8.4.1 Overview

4983 When a connection is established between Service Nodes for the transfer of IP packets, data is also  
4984 transferred in the connection request packets. This data allows the negotiation of compression and  
4985 notification of the IP address.

4986

##### 4987 5.6.8.4.2 Unicast connection data from the initiator

4988 Table 100 shows the connection data sent by the initiator.

4989

**Table 100 - IPv6 Unicast connection data sent by the initiator**

| Name            | Length   | Description         |
|-----------------|----------|---------------------|
| Data.localIPv6  | 128-bits | Local IPv6 address  |
| Data.remoteIPv6 | 128-bits | Remote IPv6 address |

4990

4991 If the device accepts the connection, it should copy both Data.localIPv6 and Data.remoteIPv6 addresses into  
4992 a new table entry , respectively in CL\_IPv6\_Con.Remote\_IP and CL\_IPv6\_Con.Local\_IP.

#### 4993 **5.6.8.4.3 Unicast connection data from the responder**

4994 No information is carried in the response's CON.DATA field.

#### 4995 **5.6.8.4.4 Multicast connection data from the initiator**

4996 **Table 101 - IPv6 Multicast connection data sent by the initiator**

| Name      | Length   | Description            |
|-----------|----------|------------------------|
| Data.IPv6 | 128-bits | IPv6 multicast address |

4997

4998 It includes only IPv6 multicast address.

#### 4999 **5.6.8.4.5 Multicast connection data from the responder**

5000 No information is carried in the response's CON.DATA field.

### 5001 **5.6.9 Service access point**

#### 5002 **5.6.9.1 Overview**

5003 This section defines the service access point used by the IPv6 layer to communicate with the IPv6  
5004 convergence layer.

#### 5005 **5.6.9.2 Opening and closing the convergence layer**

5006 The following primitives are used to open and close the convergence layer. The convergence layer may be  
5007 opened once only. The IPv6 layer may close the convergence layer when the IPv6 interface is brought down.  
5008 The convergence layer will also close the convergence layer when the underlying MAC connection to the  
5009 Base Node has been lost.

##### 5010 **5.6.9.2.1 CL\_IPv6\_Establish.request**

5011 The CL\_IPv6\_ESTABLISH.request primitive is passed from the IPv6 layer to the IPv6 convergence layer. It is  
5012 used when the IPv6 layer brings the interface up.

5013 The semantics of this primitive are as follows:

5014 *CL\_IPv6\_ESTABLISH.request{AE}*

5015 The AE parameter indicates whether the interface will be authenticated and encrypted or not.

5016 On receiving this primitive, the convergence layer will form the address resolution connection to the Base  
5017 Node.



**5018 5.6.9.2.2 CL\_IPv6\_Establish.confirm**

5019 The CL\_IPv6\_ESTABLISH.confirm primitive is passed from the IPv6 convergence layer to the IPv6 layer. It is  
5020 used to indicate that the convergence layer is ready to access IPv6 packets to be sent to peers.

5021 The semantics of this primitive are as follows:

5022 *CL\_IPv6\_ESTABLISH.confirm{AE}*

5023 The AE parameter indicates whether the interface will be authenticated and encrypted or not.

5024 Once the convergence layer has established all the necessary connections and is ready to transmit and  
5025 receive IPv6 packets, this primitive is passed to the IPv6 layer. If the convergence layer encounters an error  
5026 while opening, it responds with a CL\_IPv6\_RELEASE.confirm primitive, rather than a  
5027 CL\_IPv6\_ESTABLISH.confirm.

**5028 5.6.9.2.3 CL\_IPv6\_Release.request**

5029 The CL\_IPv6\_RELEASE.request primitive is used by the IPv6 layer when the interface is put down. The  
5030 convergence layer closes all connections so that no more IPv6 packets are received and all resources are  
5031 released.

5032 The semantics of this primitive are as follows:

5033 *CL\_IPv6\_RELEASE.request{}*

5034 Once the convergence layer has released all its connections and resources it returns a  
5035 CL\_IPv6\_RELEASE.confirm.

**5036 5.6.9.2.4 CL\_IPv6\_Release.confirm**

5037 The CL\_IPv6\_RELEASE.confirm primitive is used by the IPv6 convergence layer to indicate to the IPv6 layer  
5038 that the convergence layer has been closed. This can be as a result of a CL\_IPv6\_RELEASE.request primitive,  
5039 a CL\_IPv6\_ESTABLISH.request primitive, or because the MAC layer indicates the address resolution  
5040 connection has been lost, or the Service Node itself is no longer registered.

5041 The semantics of this primitive are as follows:

5042 *CL\_IPv6\_RELEASE.confirm{result}*

5043 The result parameter has the meanings defined in Table 153.

**5044 5.6.9.3 Unicast address management****5045 5.6.9.3.1 General**

5046 The primitives defined here are used for address management, i.e. the registration and unregistration of IPv6  
5047 addresses associated with this convergence layer.

5048 When there are no IPv6 addresses associated with the convergence layer, the convergence layer will only  
5049 send and receive multicast packets; unicast packets may not be sent. However, this is sufficient for various  
5050 address discovery protocols to be used to gain an IPv6 address. Once an IPv6 address has been registered,

5051 the IPv6 layer can transmit unicast packets that have a source address equal to one of its registered  
5052 addresses.

#### 5053 **5.6.9.3.2 CL\_IPv6\_Register.request**

5054 This primitive is passed from the IPv6 layer to the IPv6 convergence layer to register an IPv6 address.

5055 The semantics of this primitive are as follows:

5056 `CL_IPv6_REGISTER.request{ipv6, netmask, gateway}`

5057 The ipv6 address is the address to be registered.

5058 The netmask is the network mask, used to mask the network number from the address. The netmask is used  
5059 by the convergence layer to determine whether the packet should deliver directly or the gateway should be  
5060 used.

5061 The IPv6 address of the gateway, to which packets with destination address that are not in the same subnet  
5062 as the local address are to be sent.

5063 Once the IPv6 address has been registered to the Base Node, a CL\_IPv6\_REGISTER.confirm primitive is used.  
5064 If the registration fails, the CL\_IPv6\_RELEASE.confirm primitive will be used.

#### 5065 **5.6.9.3.3 CL\_IPv6\_Register.confirm**

5066 This primitive is passed from the IPv6 convergence layer to the IPv6 layer to indicate that a registration has  
5067 been successful.

5068 The semantics of this primitive are as follows:

5069 `CL_IPv6_REGISTER.confirm{ipv6}`

5070 The ipv6 address is the address that was registered.

5071 Once registration has been completed, the IPv6 layer may send IPv6 packets using this source address.

#### 5072 **5.6.9.3.4 CL\_IPv6\_Unregister.request**

5073 This primitive is passed from the IPv6 layer to the IPv6 convergence layer to unregister an IPv6 address.

5074 The semantics of this primitive are as follows:

5075 `CL_IPv6_UNREGISTER.request{ipv6}`

5076 The ipv6 address is the address to be unregistered.

5077 Once the IPv6 address has been unregistered to the Base Node, a CL\_IPv6\_UNREGISTER.confirm primitive is  
5078 used. If the registration fails, the CL\_IPv6\_RELEASE.confirm primitive will be used.

#### 5079 **5.6.9.3.5 CL\_IPv6\_Unregister.confirm**

5080 This primitive is passed from the IPv6 convergence layer to the IPv6 layer to indicate that an unregistration  
5081 has been successful.

5082 The semantics of this primitive are as follows:

5083 `CL_IPv6_UNREGISTER.confirm{ipv6}`

5084 The IPv6 address is the address that was unregistered.

5085 Once unregistration has been completed, the IPv6 layer may not send IPv6 packets using this source address.

#### 5086 **5.6.9.4 Multicast group management**

##### 5087 **5.6.9.4.1 General**

5088 This section describes the primitives used to manage multicast groups.

##### 5089 **5.6.9.4.2 CL\_IPv6\_MUL\_Join.request**

5090 This primitive is passed from the IPv6 layer to the IPv6 convergence layer. It contains an IPv6 multicast  
5091 address that is to be joined.

5092 The semantics of this primitive are as follows:

5093 `CL_IPv6_MUL_JOIN.request{IPv6 , AE }`

5094 The IPv6 address is the IPv6 multicast group that is to be joined.

5095 The AE parameter indicates whether messages in this group will be authenticated and encrypted or not.

5096 When the convergence layer receives this primitive, it will arrange for IP packets sent to this group to be  
5097 multicast in the PRIME network and receive packets using this address to be passed to the IPv6 stack. If the  
5098 convergence layer cannot join the group, it uses the CL\_IPv6\_MUL\_LEAVE.confirm primitive. Otherwise the  
5099 CL\_IPv6\_MUL\_JOIN.confirm primitive is used to indicate success.

##### 5100 **5.6.9.4.3 CL\_IPv6\_MUL\_Join.confirm**

5101 This primitive is passed from the IPv6 convergence layer to the IPv6. It contains a result status and an IPv6  
5102 multicast address that was joined.

5103 The semantics of this primitive are as follows:

5104 `CL_IPv6_MUL_JOIN.confirm{IPv6, AE}`

5105 The IPv6 address is the IPv6 multicast group that was joined. The convergence layer will start forwarding IPv6  
5106 multicast packets for the given multicast group.

5107 The AE parameter indicates whether messages in this group will be authenticated and encrypted or not.

##### 5108 **5.6.9.4.4 CL\_IPv6\_MUL\_Leave.request**

5109 This primitive is passed from the IPv6 layer to the IPv6 convergence layer. It contains an IPv6 multicast  
5110 address to be left.

5111 The semantics of this primitive are as follows:

5112 `CL_IPv6_MUL_LEAVE.request{IPv6}`

5113 The IPv6 address is the IPv6 multicast group to be left. The convergence layer will stop forwarding IPv6  
5114 multicast packets for this group and may leave the PRIME MAC multicast group.

#### 5115 **5.6.9.4.5 CL\_IPv6\_MUL\_Leave.confirm**

5116 This primitive is passed from the IPv6 convergence layer to the IPv6. It contains a result status and an IPv6  
5117 multicast address that was left.

5118 The semantics of this primitive are as follows:

5119 `CL_IPv6_MUL_LEAVE.confirm{IPv6, Result}`

5120 The IPv6 address is the IPv6 multicast group that was left. The convergence layer will stop forwarding IPv6  
5121 multicast packets for the given multicast group.

5122 The Result takes a value from Table 153.

5123 This primitive can be used by the convergence layer as a result of a CL\_IPv6\_MUL\_JOIN.request,  
5124 CL\_IPv6\_MUL\_LEAVE.request or because of an error condition resulting in the loss of the PRIME MAC  
5125 multicast connection.

#### 5126 **5.6.9.5 Data transfer**

##### 5127 **5.6.9.5.1 General**

5128 The following primitives are used to send and receive IPv6 packets.

##### 5129 **5.6.9.5.2 CL\_IPv6\_DATA.request**

5130 This primitive is passed from the IPv6 layer to the IPv6 convergence layer. It contains one IPv6 packet to be  
5131 sent.

5132 The semantics of this primitive are as follows:

5133 `CL_IPv6_DATA.request{IPv6_PDU}`

5134 The IPv6\_PDU is the IPv6 packet to be sent.

##### 5135 **5.6.9.5.3 CL\_IPv6\_DATA.confirm**

5136 This primitive is passed from the IPv6 convergence layer to the IPv6 layer. It contains a status indication and  
5137 an IPv6 packet that has just been sent.

5138 The semantics of this primitive are as follows:

5139 `CL_IPv6_DATA.confirm{IPv6_PDU, Result}`

5140 The IPv6\_PDU is the IPv6 packet that was to be sent.

5141 The Result value indicates whether the packet was sent or an error occurred. It takes a value from Table 153.

##### 5142 **5.6.9.5.4 CL\_IPv6\_DATA.indicate**

5143 This primitive is passed from the IPv6 convergence layer to the IPv6 layer. It contains an IPv6 packet that has  
5144 just been received.

5145 The semantics of this primitive are as follows:

5146       CL\_IPv6\_DATA.indicate{IPv6\_PDU }

5147 The IPv6\_PDU is the IPv6 packet that was received.

5148

## 6 Management plane

### 6.1 Introduction

This chapter specifies the Management plane functionality. The picture below highlights the position of Management plane in overall protocol architecture.

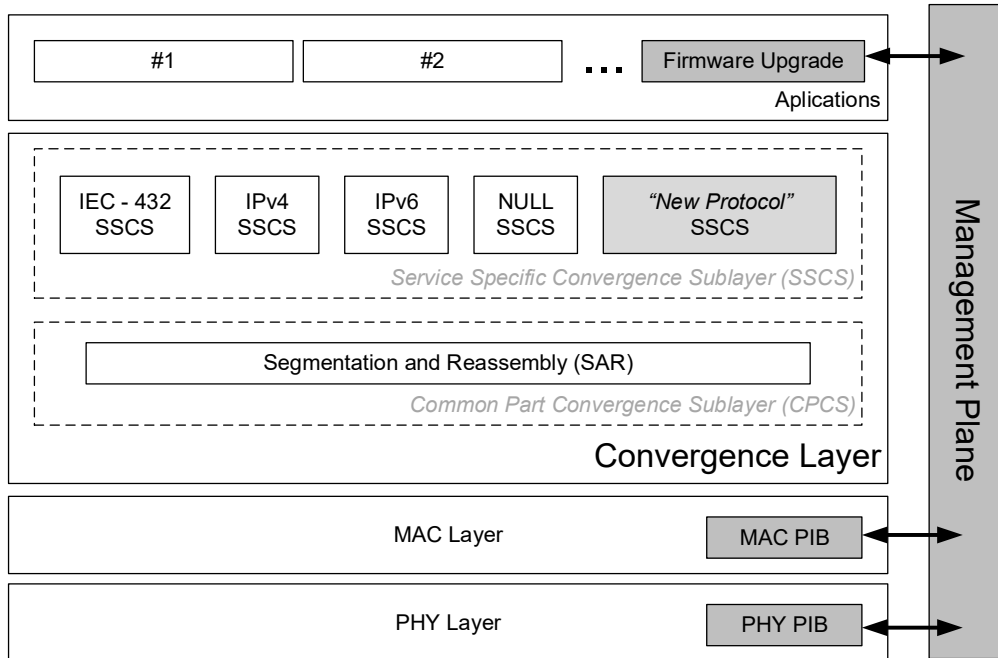


Figure 138 - Management plane. Introduction.

All nodes shall implement the management plane functionality enumerated in this section. Management plane enables a local or remote control entity to perform actions on a Node.

Present version of this specification enumerates management plane functions for Node management and firmware upgrade. Future versions may include additional management functions.

- To enable access to management functions on a Service Node, Base Node shall open a management connection after successful completion of registration (refer to 6.4)
- The Base Node may open such a connection either immediately on successful registration or sometime later.
- Unicast management connection shall be identified with CON.TYPE = TYPE\_CL\_MGMT.
- Multicast management connections can also exist. At the time of writing of this document, multicast management connection shall only be used for firmware upgrade.
- There shall be no broadcast management connection.
- In case Service Node supports ARQ connections, the Base Node shall preferentially try to open an ARQ connection for management functions.
- Management plane functions shall use NULL SSCS as specified in section 0

## 6.2 Node management

### 6.2.1 General

Node management is accomplished through a set of attributes. Attributes are defined for both PHY and MAC layers. The set of these management attributes is called PLC Information Base (PIB). Some attributes are read-only while others are read-write.

PIB Attribute identifiers are 16 bit values. This allows for up to 65535 PIB Attributes to be specified.

- PIB Attribute identifier values from 0 to 32767 are open to be standardized. No proprietary attributes may have identifiers in this range.
- Values in the range 32768 to 65535 are open for vendor specific usage.

PIB Attributes identifiers in standard range (0 to 32767) that are not specified in this version are reserved for future use.

**Note:** PIB attribute tables below indicate type of each attribute. For integer types the size of the integer has been specified in bits. An implementation may use a larger integer for an attribute; however, it must not use a smaller size.

### 6.2.2 PHY PIB attributes

#### 6.2.2.1 General

The PHY layer implementation in each device may optionally maintain a set of attributes which provide detailed information about its working. The PHY layer attributes are part of the PAN Information Base (PIB).

#### 6.2.2.2 Statistical attributes

The PHY may provide statistical information for management purposes. Next table lists the statistics that PHY should make available to management entities across the PLME\_GET primitive. The Id field in this table is the service parameter of the PLME\_GET primitive specified in section 3.5.4.

Table 102 - PHY read-only variables that provide statistical information

| Attribute Name            | Size (in bits) | Id     | Description                                                                                                                                                                                                                                                     |
|---------------------------|----------------|--------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| phyStatsCRCIncorrectCount | 16             | 0x00A0 | Number of bursts received on the PLC PHY layer for which the CRC was incorrect.                                                                                                                                                                                 |
| phyStatsCRCFailCount      | 16             | 0x00A1 | Number of bursts received on the PLC PHY layer for which the CRC was correct, but the <i>Protocol</i> field of PHY header had an invalid value. This count would reflect number of times corrupt data was received and the CRC calculation failed to detect it. |
| phyStatsTxDropCount       | 16             | 0x00A2 | Number of times when PLC PHY layer received new data to transmit (PHY_DATA.request) and had to                                                                                                                                                                  |

|                          |    |        |                                                                                                                                                                                                                                                                                                                                                                                                                            |
|--------------------------|----|--------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|                          |    |        | either overwrite on existing data in its transmit queue or drop the data in new request due to full queue.                                                                                                                                                                                                                                                                                                                 |
| phyStatsRxDropCount      | 16 | 0x00A3 | Number of times when PLC PHY layer received new data on the channel and had to either overwrite on existing data in its receive queue or drop the newly received data due to full queue.                                                                                                                                                                                                                                   |
| phyStatsRxTotalCount     | 32 | 0x00A4 | Total number of PLC PPDUs correctly decoded. Useful for PHY layer test cases, to estimate the FER.                                                                                                                                                                                                                                                                                                                         |
| phyStatsBlkAvgEvm        | 16 | 0x00A5 | Exponential moving average of the EVM over the past 16 PPDUs, as returned by the PHY_SNR primitive. Note that the PHY_SNR primitive returns a 3-bit number in dB scale. So first each 3-bit dB number is converted to linear scale (number k goes to $2^{(k/2)}$ ), yielding a 7 bit number with 3 fractional bits. The result is just accumulated over 16 PPDUs and reported. This PIB only applies to the PLC PHY layer. |
| phyEmaSmoothing          | 8  | 0x00A8 | <p>Smoothing factor divider for values that are updated as exponential moving average (EMA). Next value is</p> $V_{next} = S * NewSample + (1-S) * V_{prev}$ <p>Where</p> $S = 1 / (2^{phyEMASmoothing})$ <p>This PIB only applies to the PLC PHY layer.</p>                                                                                                                                                               |
| phyRFStatsIncorrectCount | 16 | 0x101A | Total number of RF PPDUs incorrectly decoded. Useful for PHY layer test cases, to estimate the FER.                                                                                                                                                                                                                                                                                                                        |
| phyRFStatsTxDropCount    | 16 | 0x101B | Number of times when RF PHY layer received new data to transmit (PHY_DATA.request) and had to either overwrite on existing data in its transmit queue or drop the data in new request due to full queue.                                                                                                                                                                                                                   |
| phyRFStatsRxDropCount    | 16 | 0x101C | Number of times when RF PHY layer received new data on the channel and had to either overwrite on existing data in its receive queue or drop the newly received data due to full queue.                                                                                                                                                                                                                                    |



|                        |    |        |                                                                                                       |
|------------------------|----|--------|-------------------------------------------------------------------------------------------------------|
| phyRFStatsRxTotalCount | 32 | 0x101D | Total number of RF PPDU's correctly decoded.<br>Useful for PHY layer test cases, to estimate the FER. |
|------------------------|----|--------|-------------------------------------------------------------------------------------------------------|

### 6.2.2.3 Implementation attributes

It is possible to implement PHY functions conforming to this specification in multiple ways. The multiple implementation options provide some degree of unpredictability for MAC layers. PHY implementations may optionally provide specific information on parameters which are of interest to MAC across the PLME\_GET primitive. A list of such parameters for the PLC medium which may be queried across the PLME\_GET primitives by MAC is provided in Table 103 - . All of the attributes listed in Table 103 - are implementation constants and shall not be changed.

**Table 103 - PHY read-only parameters, providing information on specific implementation**

| Attribute Name       | Size<br>(in bits) | Id     | Description                                                                                                                                                                                                                                                                                         |
|----------------------|-------------------|--------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| phyTxQueueLen        | 10                | 0x00B0 | Number of concurrent MPDUs that the PHY transmit buffers can hold.                                                                                                                                                                                                                                  |
| phyRxQueueLen        | 10                | 0x00B1 | Number of concurrent MPDUs that the PHY receive buffers can hold.                                                                                                                                                                                                                                   |
| phyTxProcessingDelay | 20                | 0x00B2 | Time elapsed from the instance when data is received on MAC-PHY communication interface to the time when it is put on the physical channel. This shall not include communication delay over the MAC-PHY interface.<br><br>Value of this attribute is in unit of microseconds.                       |
| phyRxProcessingDelay | 20                | 0x00B3 | Time elapsed from the instance when data is received on physical channel to the time when it is made available to MAC across the MAC-PHY communication interface. This shall not include communication delay over the MAC-PHY interface.<br><br>Value of this attribute is in unit of microseconds. |
| phyAgcMinGain        | 8                 | 0x00B4 | Minimum gain for the AGC $\leq$ 0dB.                                                                                                                                                                                                                                                                |
| phyAgcStepValue      | 3                 | 0x00B5 | Distance between steps in dB $\leq$ 6dB.                                                                                                                                                                                                                                                            |
| phyAgcStepNumber     | 8                 | 0x00B6 | Number of steps so that $\text{phyAgcMinGain} + (\text{phyAgcStepNumber} - 1) * \text{phyAgcStepValue} \geq 21\text{dB}$ .                                                                                                                                                                          |

#### 6.2.2.4 PHY constants of RF SUN FSK Layer

The PHY layer implementation in each device that implements SUN FSK PHY needs to include the following constants defined on clause 11.2 of IEEE 802.15.4-2015 [28]. These constants are hardware dependent and cannot be changed during operation:

**Table 104 - PHY read-only parameters of RF Layer, providing information on specific implementation**

| Attribute Name    | Id     | Description                                                                                       | Value                                                                                                                                                      |
|-------------------|--------|---------------------------------------------------------------------------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------|
| aMaxPhyPacketSize | 0x1000 | The maximum PSDU size (in octets) the PHY shall be able to receive                                | 2047 for SUN FSK PHY                                                                                                                                       |
| aTurnaroundTime   | 0x1001 | RX-to-TX or TX-to-RX turnaround time (in symbol periods), as defined in 10.2.1 and 10.2.2 of [28] | For the SUN FSK PHYs, the value is 1 ms expressed in symbol periods, rounded up to the next integer number of symbol periods using the ceiling() function. |

#### 6.2.2.5 PHY PIB attributes of RF SUN FSK Layer

The parameters described here, are defined in the clause 11.3 of IEEE 802.15.02-2015 [28]

**Table 105 - Configuration Parameters RF SUN FSK phy layer**

| Attribute Name           | Id     | Type            | Range       | Description                                                                                                                                                                                                                             |
|--------------------------|--------|-----------------|-------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| phyCurrentChannel        | 0x1010 | Integer16       | 0-511       | The value is the <i>NumChan</i> defined in 10.1.2.8 of [28], setting the RF channel to use for all following transmissions and receptions.                                                                                              |
| phyTxPower               | 0x1011 | Signed integer8 | -           | The transmit power of the device in dBm.                                                                                                                                                                                                |
| phyFskFecEnabled         | 0x1012 | Boolean         | TRUE, FALSE | A value of TRUE indicates that FEC is turned on. A value of FALSE indicates that FEC is turned off. This attribute is only valid for the SUN FSK and TVWS FSK PHY.                                                                      |
| phyFskFecInterleavingRsc | 0x1013 | Boolean         | TRUE, FALSE | A value of TRUE indicates that interleaving is enabled for RSC. A value of FALSE indicates that interleaving is disabled for RSC. This attribute is only valid for the SUN FSK and TVWS FSK PHY. Not relevant when phyFskFecScheme = 0. |

|                      |        |           |        |                                                                                                                      |
|----------------------|--------|-----------|--------|----------------------------------------------------------------------------------------------------------------------|
| phyFskFecScheme      | 0x1014 | Integer8  | 0      | A value of zero indicates that a non-recursive and non-systematic code (NRNSC) is employed, as described in Table 9. |
| phyFskPreambleLength | 0x1015 | Integer8  | 8      | The number of repetitions of the preamble pattern, as described in Table 9.                                          |
| phySunFskSfd         | 0x1016 | Integer8  | 0      | Determines which group of SFDs is used, as described in Table 9.                                                     |
| phyFskScramblePsdu   | 0x1017 | Boolean   | TRUE   | A value of TRUE indicates that data whitening of the PSDU is enabled, as described in Table 9.                       |
| phyCCADuration       | 0x1018 | Integer16 | 0-1000 | The duration for CCA, specified in symbols for the SUN PHYs, the default value duration is of 8 symbol periods.      |
| phyCCAThreshold      | 0x1019 | Integer8  | 0-100  | Number of dB above the specified receiver sensitivity for that PHY defined on 20.6.7 of [28].                        |

5211

5212 **6.2.3 MAC PIB attributes**5213 **6.2.3.1 General**

5214 **Note:** Note that the “M”(Mandatory) column in the tables below specifies if the PIB attributes are mandatory  
5215 for all devices (both Service Node and Base Node, specified as “All”), only for Service Nodes (“SN”), only for  
5216 Base Nodes (“BN”), only for BN and SN supporting SUN FSK PHY (“RF”) or not mandatory at all (“No”).

5217 **6.2.3.2 MAC variable attributes**

5218 MAC PIB variables include the set of PIB attributes that influence the functional behavior of an  
5219 implementation. These attributes may be defined external to the MAC, typically by the management entity  
5220 and implementations may allow changes to their values during normal running, i.e. even after the device  
5221 start-up sequence has been executed.

5222 An external management entity can have access to these attributes through the MLME\_GET (4.5.5.7) and  
5223 MLME\_SET (4.5.5.9) set of primitives. The Id field in the following table would be the *PIBAttribute* that needs  
5224 to be passed MLME SAP while working on these parameters

5225 **Table 106 - Table of MAC read-write variables**

| Attribute Name | Id     | Type     | M   | Valid Range | Description                                                 | Def. |
|----------------|--------|----------|-----|-------------|-------------------------------------------------------------|------|
| macVersion     | 0x0001 | Integer8 | All | 0x01        | The current MAC Version.<br>This is a ‘read-only’ attribute | 0x01 |

| Attribute Name          | Id     | Type     | M   | Valid Range     | Description                                                                                                                                                                                                     | Def. |
|-------------------------|--------|----------|-----|-----------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|------|
| macMinSwitchSearchTime  | 0x0010 | Integer8 | No  | 16 – 32 seconds | Minimum time for which a Service Node in <i>Disconnected</i> status should scan the initial Band for Beacons before it can broadcast PNPDU.<br><br>This attribute is not maintained in Base Nodes.              | 24   |
| macMaxPromotionPdu      | 0x0011 | Integer8 | No  | 1 – 4           | Maximum number of PNPDU's that may be transmitted by a Service Node in a period of <i>macPromotionPduTxPeriod</i> seconds.<br><br>This attribute is not maintained in Base Node.                                | 2    |
| macPromotionPduTxPeriod | 0x0012 | Integer8 | No  | 2 – 8 seconds   | Time quantum for limiting a number of PNPDU's transmitted from a Service Node. No more than <i>macMaxPromotionPdu</i> may be transmitted in a period of <i>macPromotionPduTxPeriod</i> seconds.                 | 5    |
| macSCPMaxTxAttempts     | 0x0014 | Integer8 | No  | 2 – 5           | Number of times the PLC CSMA algorithm would attempt to transmit requested data when a previous attempt was withheld due to PHY indicating channel busy.                                                        | 5    |
| macMinCtlReTxTimer      | 0x0015 | Integer8 | All | 2 sec           | Minimum number of seconds for which a MAC entity waits for acknowledgement of receipt of MAC Control Packet from its peer entity. On expiry of this time, the MAC entity may retransmit the MAC Control Packet. | 2    |

| Attribute Name          | Id     | Type     | M   | Valid Range | Description                                                                                                                                                                                                 | Def. |
|-------------------------|--------|----------|-----|-------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|------|
| macTrafficBandTimeout   | 0x0016 | Integer8 | No  | 32 – 120    | Period of time in seconds for which a Disconnected Node shall listen on a specific PLC band before moving to another one when traffic has been detected in current PLC band.                                | 60   |
| macCtrlMsgFailTime      | 0x0018 | Integer8 | No  | 6 - 100     | Number of seconds for which a MAC entity in Switch Nodes waits before declaring a children's transaction procedures expired                                                                                 | 45   |
| macEMASmoothing         | 0x0019 | Integer8 | All | 0 - 7       | <p>Smoothing factor divider for values that are updated as exponential moving average (EMA). Next value is</p> $V_{next} = S * NewSample + (1 - S) * V_{prev}$ <p>Where</p> $S = 1 / (2^{macEMASmoothing})$ | 3    |
| macMinBandSearchTime    | 0x001A | Integer8 | No  | 4 - 32      | Period of time in seconds for which a Disconnected Node shall listen on a specific PLC band before moving to another one when traffic has not been detected in current PLC band.                            | 10   |
| macPromotionMaxTxPeriod | 0x001B | Integer8 | SN  | 16-120      | <p>Period of time in seconds for which at least one PNPDU shall be sent</p> <p>Note: This attribute is deprecated in v1.4 and only maintained by devices implementing v1.3.6</p>                            | 32   |

| Attribute Name          | Id     | Type      | M   | Valid Range | Description                                                                                                                                                                                                                                                                                                                                                     | Def. |
|-------------------------|--------|-----------|-----|-------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|------|
| macPromotionMinTxPeriod | 0x001C | Integer8  | SN  | 2-16        | Period of time in seconds for which at no more than one PNPDU shall be sent.                                                                                                                                                                                                                                                                                    | 2    |
| macSARSize              | 0x001D | Integer8  | All | 0-7         | <p>Maximum Data packet segment size.</p> <p>0: Not mandated by BN (SAR operates normally)</p> <p>1: SAR = 16 bytes</p> <p>2: SAR =32 bytes</p> <p>3: SAR = 48 bytes</p> <p>4: SAR =64 bytes</p> <p>5: SAR =128 bytes</p> <p>6: SAR =192 bytes</p> <p>7: SAR =255 bytes</p> <p>This attribute can be modified only in Base Node. Read-only for Service Nodes</p> | 0    |
| macMaxBandSearchTime    | 0x001E | Integer16 | No  | 300 – 3600  | Maximum period of time in seconds for which a Disconnected Node shall listen on a specific PLC band before moving to another.                                                                                                                                                                                                                                   | 1800 |
| macRobustnessManagement | 0x004A | Integer8  | No  | 0-3         | <p>Force the network to operate only with one specific modulation</p> <p>0 – No forcing automatic robustness-management</p> <p>1 – Use only DBPSK_CC</p> <p>2 - Use only DQPSK_R</p> <p>3 - Use only DBPSK_R</p> <p>This attribute can be modified only in Base Node. Read-only for Service Nodes</p>                                                           | 0    |

| Attribute Name                    | Id     | Type        | M   | Valid Range   | Description                                                                                                                                                                                   | Def.                                                                                                                                                                                                |
|-----------------------------------|--------|-------------|-----|---------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| macUpdatedRMTimeout               | 0x004B | Integer16   | All | 60-3600       | Period of time in seconds for which an entry in the                                                                                                                                           | 240                                                                                                                                                                                                 |
| macALVHopRepetitions              | 0x004C | Integer8    | All | 0-7           | Number of repletion for the ALV packets                                                                                                                                                       | 5                                                                                                                                                                                                   |
| macPhyChannelChange               | 0x004D | 3x Integer8 | No  |               | Change the Physical layer channel/band used by Base Node on a specific medium and notify the event through a PCC MAC control packet<br><br>This attribute is not maintained in Service Nodes. | -                                                                                                                                                                                                   |
|                                   |        |             |     | Entry Element | Size                                                                                                                                                                                          | Description                                                                                                                                                                                         |
|                                   |        |             |     | PCH           | Integer16                                                                                                                                                                                     | New Physical layer channel/band that will be used by the Base Node on a specific medium, as described in 4.4.2.6.11. The 10 least significant bits of the entry element are used to store the value |
|                                   |        |             |     | SeqOffset     | Integer8                                                                                                                                                                                      | An offset, to be added to current Beacon Sequence number, used to select the frame when the specified change takes effect (range 1-31)                                                              |
| macPhyPLCNetworkAttenuationChange | 0x004F | 2x Integer8 | PLC |               | Change the PLC network attenuation and notify the event through a PCC MAC control packet<br><br>This attribute is not maintained in Service Nodes.                                            | -                                                                                                                                                                                                   |

| Attribute Name                 | Id     | Type     | M             | Valid Range  | Description                                                                                                                                                                                                                         | Def. |
|--------------------------------|--------|----------|---------------|--------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|------|
|                                |        |          | Entry Element | Size         | Description                                                                                                                                                                                                                         |      |
|                                |        |          | PLC_ATT       | Integer8     | New PLC network, as described in 4.4.2.6.11.                                                                                                                                                                                        |      |
|                                |        |          | SeqOffset     | Integer8     | An offset, to be added to current Beacon Sequence number, used to select the frame when the specified change takes effect (range 1-31)                                                                                              |      |
| macMinBe                       | 0x0098 | Integer8 | RF            | 0 - macMaxBe | Description in Table 8-81 of [28]                                                                                                                                                                                                   | 3    |
| macMaxBe                       | 0x0099 | Integer8 | RF            | 3 - 8        | Description in Table 8-81 of [28]                                                                                                                                                                                                   | 5    |
| macMaxCsmaBackoffs             | 0x009A | Integer8 | RF            | 0-5          | Description in Table 8-81 of [28]                                                                                                                                                                                                   | 4    |
| macHoppingPromotionMaxTxPeriod | 0x009B | Integer8 | RF SN         | 16-120       | Period of time in seconds for which at least one PNPDU per RF medium channel shall be sent.                                                                                                                                         | 32   |
| macHoppingPromotionMinTxPeriod | 0x009C | Integer8 | RF SN         | 2-16         | Period of time in seconds for which at no more than one PNPDU per RF medium channel shall be sent.                                                                                                                                  | 2    |
| macHoppingInitialChannelList   | 0x009D | 512 bits | RF            | bitmap       | RF channels used to generate the main hopping sequence. Representing macHoppingInitialChannelList by $b_{511}b_{510}...b_2b_1b_0$ , $b_k=1$ indicates that channel $k$ is used. $b_0$ is the lsb and byte convention is big endian. | -    |



| Attribute Name                  | Id     | Type      | M  | Valid Range | Description                                                                                                                                                                                                                            | Def. |
|---------------------------------|--------|-----------|----|-------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|------|
| macHoppingBCNInitialChannelList | 0x009E | 512 bits  | RF | bitmap      | RF channels used to generate the beacon hopping sequence. Representing macHoppingBCNInitialChannelList by $b_{511}b_{510}...b_2b_1b_0$ , $b_k=1$ indicates that channel k is used. $b_0$ is the lsb and byte convention is big endian. | -    |
| macTrafficRFChannelTimeout      | 0x00C0 | Integer8  | No | 32 – 120    | Period of time in seconds for which a Disconnected Node shall listen on a specific RF channel before moving to another one when traffic has been detected in current RF channel.                                                       | 60   |
| macMinRFChannelSearchTime       | 0x00C1 | Integer8  | No | 4 - 32      | Period of time in seconds for which a Disconnected Node shall listen on a specific RF channel before moving to another one when traffic has not been detected in current RF channel.                                                   | 10   |
| macMaxRFChannelSearchTime       | 0x00C2 | Integer16 | No | 300 – 3600  | Maximum period of time in seconds for which a Disconnected Node shall listen on a specific RF channel before moving to another.                                                                                                        | 1800 |
| macRFLinkCost                   | 0x00C3 | Integer8  | RF | 0 - 8       | Network pre-shared RF link cost to be adopted in the computation of BCN.COST (see Table 42).                                                                                                                                           | 0    |

Table 107 - Table of MAC read-only variables

| Attribute Name     | Id     | Type     | M   | Valid Range | Description                                                                                                                                                      | Def  |
|--------------------|--------|----------|-----|-------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------|------|
| macSCPChSenseCount | 0x0017 | Integer8 | No  | 2 – 5       | Number of times for which an implementation has to perform channel-sensing.<br><br>This is a 'read-only' attribute.                                              | -    |
| macEUI-48          | 0x001F | EUI-48   | All |             | EUI-48 of the Node                                                                                                                                               | -    |
| macCSMAR1          | 0x0034 | Integer8 | All | 0 - 4       | Control how fast the CSMA contention window shall increase. Controls exponential increase of initial CSMA contention window size                                 | 3    |
| macCSMAR2          | 0x0035 | Integer8 | All | 1 - 4       | Control initial CSMA contention window size. Controls linear increase of initial CSMA contention window size.                                                    | 1    |
| macCSMADelay       | 0x0038 | Integer8 | All | 3ms – 9ms   | The delay between two consecutive CSMA channel senses.                                                                                                           | 3 ms |
| macCSMAR1Robust    | 0x003B | Integer8 | All | 0 - 5       | Control how fast the CSMA contention window shall increase when node supports Robust Mode. Controls exponential increase of initial CSMA contention window size. | 4    |
| macCSMAR2Robust    | 0x003C | Integer8 | All | 1 - 8       | Control initial CSMA contention window size when node supports Robust Mode. Controls linear increase of initial CSMA contention window size.                     | 2    |
| macCSMADelayRobust | 0x003D | Integer8 | All | 3ms – 9ms   | The delay between two consecutive CSMA channel senses when node supports Robust Mode.                                                                            | 6 ms |

| Attribute Name                | Id     | Type      | M   | Valid Range   | Description                                                                                                                                                                    | Def |
|-------------------------------|--------|-----------|-----|---------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-----|
| macAliveTimeMode              | 0x003E | Integer8  | BN  | 0 - 1         | Selects the <i>MACAliveTime</i> value mapping for network Alive time.<br><br>0 – 1.4 mode<br>1 – BC mode                                                                       | -   |
| macPhyPLCNetworkAttenuation   | 0x004E | Integer8  | PLC | 0-31          | PLC network attenuation currently in use by the node.<br><br>This is a 'read-only' attribute.                                                                                  | 0   |
| macNumberOfRFChannels         | 0x0090 | Integer16 | RF  | 0x0000-0x0200 | Number of channels supported by the RF PHY.                                                                                                                                    | -   |
| macHoppingSequenceLength      | 0x0091 | Integer16 | RF  | 0x0000-0x0200 | The number of channels in the hopping sequence. It can be lower than or equal to macNumberOfRFChannels. It is equal to the number of "1" bits in macHoppingInitialChannelList. | -   |
| macHoppingSequencePosition    | 0x0092 | Integer16 | No  | 0x0000-0x01ff | The current position in the main sequence for RF channel hopping.                                                                                                              | -   |
| macHoppingBCNSequenceLength   | 0x0093 | Integer8  | RF  | 0x00-0x20     | The number of channels in the beacon hopping sequence. It is equal to the number of "1" bits in macHoppingBCNInitialChannelList                                                | -   |
| macHoppingBCNSequencePosition | 0x0094 | Integer8  | No  | 0x00-0x1f     | The current position in the beacon sequence for RF channel hopping.                                                                                                            | -   |

5228

### 5229 6.2.3.3 Functional attributes

5230 Some PIB attributes belong to the functional behavior of MAC. They provide information on specific aspects.  
 5231 A management entity can only read their present value using the MLME\_GET primitives. The value of these  
 5232 attributes cannot be changed by a management entity through the MLME\_SET primitives.

5233 The Id field in the table below would be the *PIBAttribute* that needs to be passed MLME\_GET SAP for  
 5234 accessing the value of these attributes.

5235 **Table 108 - Table of MAC read-only variables that provide functional information**

| Attribute Name        | Id     | Type      | M  | Valid Range | Description                                                                                                                                                                                                                                                              |
|-----------------------|--------|-----------|----|-------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| macLNID               | 0x0020 | Integer16 | SN | 0 – 16383   | LNID allocated to this Node at time of its registration. (0x0000 is reserved for Base Node)                                                                                                                                                                              |
| macLSID               | 0x0021 | Integer8  | SN | 0 – 255     | LSID allocated to this Node at time of its promotion. This attribute is not maintained if a Node is in a <i>Terminal</i> functional state. (0x00 is reserved for Base Node). This attribute is maintained only for a promotion accomplished using the PRO_REQ_x message. |
| macSID                | 0x0022 | Integer8  | SN | 0 – 255     | SID of the Switch Node through which this Node is connected to the Subnetwork. This attribute is not maintained in a Base Node.                                                                                                                                          |
| macSNA                | 0x0023 | EUI-48    | SN |             | Subnetwork address to which this Node is registered.<br><br>The Base Node returns the SNA it is using.                                                                                                                                                                   |
| macState              | 0x0024 | Enumerate | SN |             | Present functional state of the Node.                                                                                                                                                                                                                                    |
|                       |        |           |    | 0           | DISCONNECTED.                                                                                                                                                                                                                                                            |
|                       |        |           |    | 1           | TERMINAL.                                                                                                                                                                                                                                                                |
|                       |        |           |    | 2           | SWITCH.                                                                                                                                                                                                                                                                  |
|                       |        |           |    | 3           | BASE.                                                                                                                                                                                                                                                                    |
| macSCPLength          | 0x0025 | Integer16 | SN |             | The SCP length, in symbols, in present frame.                                                                                                                                                                                                                            |
| macNodeHierarchyLevel | 0x0026 | Integer8  | SN | 0 – 63      | Level of this Node in Subnetwork hierarchy.                                                                                                                                                                                                                              |

| Attribute Name       | Id     | Type      | M  | Valid Range | Description                                                                                                                                                                                                                                                                                                                                                                         |
|----------------------|--------|-----------|----|-------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| macBeaconRxPos       | 0x0039 | Integer16 | SN | 0 – 1104    | Beacon Position on which this device's Switch Node transmits its beacon. Position is expressed in terms of symbols from the start of the frame. This attribute is not maintained in a Base Node.                                                                                                                                                                                    |
| macBeaconTxPos       | 0x003A | Integer8  | SN | 0 – 1104    | Beacon Position in which this device transmits its beacon. Position is expressed in terms of symbols from the start of the frame. This attribute is not maintained in Service Nodes that are in a <i>Terminal</i> functional state. This attribute is maintained only for a promotion accomplished using the PRO_REQ_x message.                                                     |
| macBeaconRxFrequency | 0x002A | Integer8  | SN | 0 – 5       | Number of frames between receptions of two successive beacons. A value of 0x0 indicates beacons are received in every frame. This attribute is not maintained in Base Node. Use the same encoding of FRQ field in the packets                                                                                                                                                       |
| macBeaconTxFrequency | 0x002B | Integer8  | SN | 0 – 5       | Number of frames between transmissions of two successive beacons. A value of 0x0 indicates beacons are transmitted in every frame. This attribute is not maintained in Service Nodes that are in a <i>Terminal</i> functional state. Use the same encoding of FRQ field in the packets. This attribute is maintained only for a promotion accomplished using the PRO_REQ_x message. |

| Attribute Name  | Id     | Type      | M   | Valid Range | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             |
|-----------------|--------|-----------|-----|-------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| macCapabilities | 0x002C | Integer16 | All | Bitmap      | <p>Bitmap of MAC capabilities of a given device. This attribute shall be maintained on all devices. Bits in sequence of right-to-left shall have the following meaning:</p> <p><b>Bit0:</b> Robust mode Capable;</p> <p><b>Bit1:</b> Backward Compatible Capable;</p> <p><b>Bit2:</b> Switch Capable;</p> <p><b>Bit3:</b> Packet Aggregation Capable;</p> <p><b>Bit4:</b> Connection Free Period Capable;</p> <p><b>Bit5:</b> Direct Connection Capable;</p> <p><b>Bit6:</b> ARQ Capable;</p> <p><b>Bit7:</b> Multi-PHY Promotion Capable;</p> <p><b>Bit8:</b> Direct Connection Switching;</p> <p><b>Bit9:</b> Multicast Switching Capability;</p> <p><b>Bit10:</b> Robust promotion device Capable;</p> <p><b>Bit11:</b> ARQ Buffering Switching Capability;</p> <p><b>Bits12 to 15:</b> Reserved for future use.</p> |
| macFrameLength  | 0x002D | Integer16 | All | 0 – 3       | <p>Frame Length in the present super-frame</p> <p>0 - 276 symbols</p> <p>1 - 552 symbols</p> <p>2 - 828 symbols</p> <p>3 - 1104 symbols</p>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             |
| macCFPLength    | 0x002E | Integer16 | All |             | The CFP length in symbols, in present frame                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             |

| Attribute Name  | Id     | Type      | M   | Valid Range | Description                                                           |
|-----------------|--------|-----------|-----|-------------|-----------------------------------------------------------------------|
| macGuardTime    | 0x002F | Integer16 | All | 1 symbol    | The guard time between portion of the frame in symbols                |
| macBCMode       | 0x0030 | Integer16 | All | 0 or 1      | MAC is operating in Backward Compatibility Mode                       |
| macBeaconRxQlty | 0x0032 | Integer16 | All |             | The BCN.QLTY field this device's Switch Node transmits in its beacon. |
| macBeaconTxQlty | 0x0033 | Integer16 | All |             | The BCN.QLTY field transmitted by this device in its beacon.          |

5236

5237 **6.2.3.4 Statistical attributes**

5238 The MAC layer shall provide statistical information for management purposes. Table 109 lists the statistics  
 5239 MAC shall make available to management entities across the MLME\_GET primitive.

5240 The Id field in table below would be the *PIBAttribute* that needs to be passed MLME\_GET SAP for accessing  
 5241 the value of these attributes.

5242 **Table 109 - Table of MAC read-only variables that provide statistical information**

| Attribute Name      | Id     | M  | Type      | Description                                                                                                          |
|---------------------|--------|----|-----------|----------------------------------------------------------------------------------------------------------------------|
| macTxDataPktCount   | 0x0040 | No | Integer32 | Count of successfully transmitted MSDUs.                                                                             |
| MacRxDataPktCount   | 0x0041 | No | Integer32 | Count of successfully received MSDUs whose destination address was this Node.                                        |
| MacTxCtrlPktCount   | 0x0042 | No | Integer32 | Count of successfully transmitted MAC control packets.                                                               |
| MacRxCtrlPktCount   | 0x0043 | No | Integer32 | Count of successfully received MAC control packets whose destination address was this Node.                          |
| MacCSMAFailCount    | 0x0044 | No | Integer32 | Count of failed PLC CSMA transmitted attempts.                                                                       |
| MacCSMAChBusyCount  | 0x0045 | No | Integer32 | Count of number of times this Node had to back off SCP transmission due to channel busy state, when PLC CSMA is used |
| MacRFCSMACFailCount | 0x0046 | No | Integer32 | Count of failed RF CSMA transmitted attempts.                                                                        |

| Attribute Name       | Id     | M  | Type      | Description                                                                                                         |
|----------------------|--------|----|-----------|---------------------------------------------------------------------------------------------------------------------|
| MacRFCSMAChBusyCount | 0x0047 | No | Integer32 | Count of number of times this Node had to back off SCP transmission due to channel busy state, when RF CSMA is used |
| MacRxBcnPktCount     | 0x0048 | No | Integer32 | Count of successfully received beacon packets.                                                                      |

### 6.2.3.5 MAC list attributes

MAC layer shall make certain lists available to the management entity across the MLME\_LIST\_GET primitive. These lists are given in Table 110. Although a management entity can read each of these lists, it cannot change the contents of any of them.

The Id field in table below would be the *PIBListAttribute* that needs to be passed MLME\_LIST\_GET primitive for accessing the value of these attributes.

**Table 110 - Table of read-only lists made available by MAC layer through management interface**

| List Attribute Name | Id     | M  | Description                                                                                                                                                                                                                                                  |                          |                                                                                                                                             |
|---------------------|--------|----|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|--------------------------|---------------------------------------------------------------------------------------------------------------------------------------------|
| macListRegDevices   | 0x0050 | BN | List of registered devices. This list is maintained by the Base Node only. Each entry in this list shall comprise the following information. Note that this list is extended in 6.2.3.5.1 to better address the case where Multi-PHY promotion is supported. |                          |                                                                                                                                             |
|                     |        |    | Entry Element                                                                                                                                                                                                                                                | Type                     | Description                                                                                                                                 |
|                     |        |    | regEntryID                                                                                                                                                                                                                                                   | EUI-48                   | EUI-48 of the registered Node.                                                                                                              |
|                     |        |    | regEntryLNID                                                                                                                                                                                                                                                 | Integer16                | LNID allocated to this Node.                                                                                                                |
|                     |        |    | regEntryState                                                                                                                                                                                                                                                | TERMINAL=1<br>, SWITCH=2 | Functional state of this Node.                                                                                                              |
|                     |        |    | regEntryLSID                                                                                                                                                                                                                                                 | Integer8                 | SID allocated to this Node. This entry element corresponds to a PRO.NSID obtained in a promotion accomplished using the PRO_REQ_x messages. |
|                     |        |    | regEntrySID                                                                                                                                                                                                                                                  | Integer8                 | SID of Switch through which this Node is connected.                                                                                         |
|                     |        |    | regEntryLevel                                                                                                                                                                                                                                                | Integer8                 | Hierarchy level of this Node.                                                                                                               |



| List Attribute Name | Id     | M  | Description                                                                           |          |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |
|---------------------|--------|----|---------------------------------------------------------------------------------------|----------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|                     |        |    | regEntryTCap                                                                          | Integer8 | <p>Bitmap of MAC Capabilities of Terminal functions in this device.</p> <p>Bits in sequence of right-to-left shall have the following meaning:</p> <p><b>Bit0:</b> Robust mode Capable;<br/> <b>Bit1:</b> Backward Compatible Capable;<br/> <b>Bit2:</b> Switch Capable;<br/> <b>Bit3:</b> Packet Aggregation Capable;<br/> <b>Bit4:</b> Connection Free Period Capable;<br/> <b>Bit5:</b> Direct Connection Capable;<br/> <b>Bit6:</b> ARQ Capable;<br/> <b>Bit7:</b> Multi-PHY Promotion Capable.</p> |
|                     |        |    | regEntrySwCap                                                                         | Integer8 | <p>Bitmap of MAC Switching capabilities of this device</p> <p>Bits in sequence of right-to-left shall have the following meaning:</p> <p><b>Bit0:</b> Direct Connection Switching;<br/> <b>Bit1:</b> Reserved;<br/> <b>Bit2:</b> Reserved;<br/> <b>Bit3:</b> ARQ Buffering Switching Capability;<br/> <b>Bit4 to 7:</b> Reserved for future use.</p>                                                                                                                                                    |
| macListActiveConn   | 0x0051 | BN | List of active non-direct connections. This list is maintained by the Base Node only. |          |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |
|                     |        |    | Entry Element                                                                         | Type     | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             |

| List Attribute Name | Id     | M  | Description                                                                                                                       |           |                                                                                   |
|---------------------|--------|----|-----------------------------------------------------------------------------------------------------------------------------------|-----------|-----------------------------------------------------------------------------------|
|                     |        |    | connEntrySID                                                                                                                      | Integer8  | SID of Switch through which the Service Node is connected.                        |
|                     |        |    | connEntryLNID                                                                                                                     | Integer16 | NID allocated to Service Node.                                                    |
|                     |        |    | connEntryLCID                                                                                                                     | Integer16 | LCID allocated to this connection.                                                |
|                     |        |    | connEntryID                                                                                                                       | EUI-48    | EUI-48 of Service Node.                                                           |
| macListMcastEntries | 0x0052 | No | List of entries in multicast switching table. This list is not maintained by Service Nodes in a <i>Terminal</i> functional state. |           |                                                                                   |
|                     |        |    | Entry Element                                                                                                                     | Type      | Description                                                                       |
|                     |        |    | mcastEntryLCID                                                                                                                    | Integer16 | LCID of the multicast group.                                                      |
|                     |        |    | mcastEntryMembers                                                                                                                 | Integer16 | Number of child Nodes (including the Node itself) that are members of this group. |
| macListSwitchTable  | 0x005A | SN | List the Switch table. This list is not maintained by Service Nodes in a <i>Terminal</i> functional state.                        |           |                                                                                   |
|                     |        |    | Entry Element                                                                                                                     | Type      | Description                                                                       |
|                     |        |    | stblEntryLNID                                                                                                                     | Integer16 | LNID of attached Switch Node.                                                     |
|                     |        |    | stblEntryLSID                                                                                                                     | Integer8  | LSID assigned to the attached Switch Node.                                        |
|                     |        |    | stblEntrySID                                                                                                                      | Integer8  | SID of attached Switch Node                                                       |
|                     |        |    | stblEntryALVTime                                                                                                                  | Integer8  | The TIME value used for the Keep Alive process                                    |
| macListDirectConn   | 0x0054 | No | List of direct connections that are active. This list is maintained only in the Base Node.                                        |           |                                                                                   |
|                     |        |    | Entry Element                                                                                                                     | Type      | Description                                                                       |
|                     |        |    | dconnEntrySrcSID                                                                                                                  | Integer8  | SID of Switch through which the source Service Node is connected.                 |
|                     |        |    | dconEntrySrcLNID                                                                                                                  | Integer16 | NID allocated to the source Service Node.                                         |

| List Attribute Name | Id     | M  | Description                  |           |                                                                        |
|---------------------|--------|----|------------------------------|-----------|------------------------------------------------------------------------|
|                     |        |    | dconnEntrySrcLCID            | Integer16 | LCID allocated to this connection at the source.                       |
|                     |        |    | dconnEntrySrcID              | EUI-48    | EUI-48 of source Service Node.                                         |
|                     |        |    | dconnEntryDstSID             | Integer8  | SID of Switch through which the destination Service Node is connected. |
|                     |        |    | dconnEntryDstLNID            | Integer16 | NID allocated to the destination Service Node.                         |
|                     |        |    | dconnEntryDstLCID            | Integer16 | LCID allocated to this connection at the destination.                  |
|                     |        |    | dconnEntryDstID              | EUI-48    | EUI-48 of destination Service Node.                                    |
|                     |        |    | dconnEntryDSID               | Integer8  | SID of Switch that is the direct Switch.                               |
|                     |        |    | dconnEntryDID                | EUI-48    | EUI-48 of direct switch.                                               |
| macListDirectTable  | 0x0055 | No | List the direct Switch table |           |                                                                        |
|                     |        |    | Entry Element                | Type      | Description                                                            |
|                     |        |    | dconnEntrySrcSID             | Integer8  | SID of Switch through which the source Service Node is connected.      |
|                     |        |    | dconEntrySrcLNID             | Integer16 | NID allocated to the source Service Node.                              |
|                     |        |    | dconnEntrySrcLCID            | Integer16 | LCID allocated to this connection at the source.                       |
|                     |        |    | dconnEntryDstSID             | Integer8  | SID of Switch through which the destination Service Node is connected. |
|                     |        |    | dconnEntryDstLNID            | Integer16 | NID allocated to the destination Service Node.                         |

| List Attribute Name      | Id     | M  | Description                                                                                                                                                      |                 |                                                                                                                                           |
|--------------------------|--------|----|------------------------------------------------------------------------------------------------------------------------------------------------------------------|-----------------|-------------------------------------------------------------------------------------------------------------------------------------------|
|                          |        |    | dconnEntryDstLCID                                                                                                                                                | Integer16       | LCID allocated to this connection at the destination.                                                                                     |
|                          |        |    | dconnEntryDID                                                                                                                                                    | EUI-48          | EUI-48 of direct switch.                                                                                                                  |
| macListAvailableSwitches | 0x0056 | SN | List of Switch Nodes whose beacons are received. Note that this list is extended in 6.2.3.5.1 to better address the case where Multi-PHY promotion is supported. |                 |                                                                                                                                           |
|                          |        |    | <b>Entry Element</b>                                                                                                                                             | <b>Type</b>     | <b>Description</b>                                                                                                                        |
|                          |        |    | slistEntrySNA                                                                                                                                                    | EUI-48          | EUI-48 of the Subnetwork.                                                                                                                 |
|                          |        |    | slistEntryLSID                                                                                                                                                   | Integer8        | SID of this Switch.                                                                                                                       |
|                          |        |    | slistEntryLevel                                                                                                                                                  | Integer8        | Level of this Switch in Subnetwork hierarchy.                                                                                             |
|                          |        |    | slistEntryRxLvl                                                                                                                                                  | Integer8<br>EMA | Received signal level for this Switch. EMA output is determined using values that correspond to the Level parameter defined in 3.5.2.4.2. |
|                          |        |    | slistEntryRxSNR                                                                                                                                                  | Integer8<br>EMA | Signal to Noise Ratio for this Switch. EMA output is determined using values that correspond to the SNR parameter defined in 3.5.3.12.2.  |
|                          | 0x0057 |    | Deprecated since v1.3.6 of specs and reserved for future use                                                                                                     |                 |                                                                                                                                           |
| macListActiveConnections | 0x0058 | BN | List of active non-direct connections. This list is maintained by the Base Node only. Extended version.                                                          |                 |                                                                                                                                           |
|                          |        |    | <b>Entry Element</b>                                                                                                                                             | <b>Type</b>     | <b>Description</b>                                                                                                                        |
|                          |        |    | connEntrySID                                                                                                                                                     | Integer16       | SID of Switch through which the Service Node is connected.                                                                                |
|                          |        |    | connEntryLNID                                                                                                                                                    | Integer16       | NID allocated to Service Node.                                                                                                            |

| List Attribute Name | Id     | M   | Description                                                                                                                                                                                                                                                                                                                                                                 |                 |                                                                                                                                                             |
|---------------------|--------|-----|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-----------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------|
|                     |        |     | connEntryLCID                                                                                                                                                                                                                                                                                                                                                               | Integer16       | LCID allocated to this connection.                                                                                                                          |
|                     |        |     | connEntryID                                                                                                                                                                                                                                                                                                                                                                 | EUI-48          | EUI-48 of Service Node.                                                                                                                                     |
|                     |        |     | connType                                                                                                                                                                                                                                                                                                                                                                    | Integer8        | Type of connection.                                                                                                                                         |
| macListPhyComm      | 0x0059 | All | List of PHY communication parameters. This table is maintained in every Node. For Terminal Nodes it contains only one entry for the Switch the Node is connected through. For other Nodes it contains also entries for every directly connected child Node. Note that this list is extended in 6.2.3.5.1 to better address the case where Multi-PHY promotion is supported. |                 |                                                                                                                                                             |
|                     |        |     | Entry Element                                                                                                                                                                                                                                                                                                                                                               | Type            | Description                                                                                                                                                 |
|                     |        |     | phyCommLNID                                                                                                                                                                                                                                                                                                                                                                 | Integer16       | LNID of the peer device                                                                                                                                     |
|                     |        |     | phyCommSID                                                                                                                                                                                                                                                                                                                                                                  | Integer8        | SID of the peer device                                                                                                                                      |
|                     |        |     | phyCommTxPwr                                                                                                                                                                                                                                                                                                                                                                | Integer8        | Tx power of GPDU packets sent to the device.                                                                                                                |
|                     |        |     | phyCommRxLvl                                                                                                                                                                                                                                                                                                                                                                | Integer8<br>EMA | Rx power level of GPDU packets received from the device. EMA output is determined using values that correspond to the Level parameter defined in 3.5.2.4.2. |
|                     |        |     | phyCommSNR                                                                                                                                                                                                                                                                                                                                                                  | Integer8<br>EMA | SNR of GPDU packets received from the device. EMA output is determined using values that correspond to the SNR parameter defined in 3.5.3.12.2.             |

| List Attribute Name | Id | M | Description              |           |                                                                                                                                                                                               |
|---------------------|----|---|--------------------------|-----------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|                     |    |   | phyCommTxModulation      | Integer8  | For PLC, modulation scheme to be used for communicating with this node.<br><br>For RF this field is reserved for future use.                                                                  |
|                     |    |   | phyCommPhyTypeCapability | Integer8  | For PLC, capability of the node to receive only PHY Type A or PHY Type A+B frames<br><br>0: Type A only node<br>1: Type A+B capable node<br><br>For RF this field is reserved for future use. |
|                     |    |   | phyCommRxAge             | Integer16 | For PLC, time [seconds] since last update of phyCommTxModulation.<br><br>For RF this field is reserved for future use.                                                                        |

#### 5250 6.2.3.5.1 Extension for Multi-PHY MAC list attributes

5251 In addition to the lists reported in 6.2.3.5, when Multi-PHY promotion is supported, the following lists are  
5252 also supported.

5253 Table 111 - Read-only lists only available for Multi-PHY extension

| List Attribute Name | Id     | M  | Description                                                                                                                                                                                                                                                                |          |                                                       |
|---------------------|--------|----|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|----------|-------------------------------------------------------|
| macListMPSwitches   | 0x2000 | SN | List with the characteristics when a Service Node is a switch in different mediums. This list is not maintained by Service Nodes in a <i>Terminal</i> functional state and it is associated to promotions accomplished using PRO_REQ_x and/or PRO_REQ_x_MultiPHY messages. |          |                                                       |
|                     |        |    | Entry Element                                                                                                                                                                                                                                                              | Type     | Description                                           |
|                     |        |    | switchLSID                                                                                                                                                                                                                                                                 | Integer8 | LSID allocated to this Node at time of its promotion. |

| List Attribute Name | Id     | M  | Description                                                                                                                                  |                         |                                                                                                                                                                                       |
|---------------------|--------|----|----------------------------------------------------------------------------------------------------------------------------------------------|-------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|                     |        |    | switchBeaconTxPos                                                                                                                            | Integer16               | Beacon Position in which this Switch transmits its beacon. Position is expressed in terms of symbols from the start of the frame.                                                     |
|                     |        |    | switchBeaconTxFrequency                                                                                                                      | Integer8                | Number of frames between transmissions of two successive beacons. A value of 0x0 indicates beacons are transmitted in every frame. Use the same encoding of FRQ field in the packets. |
|                     |        |    | switchPCH                                                                                                                                    | Integer16               | The medium and the band (PLC) or channel (RF) corresponding to the allocated LSID. This field is coded as the PRO.PCH field of Table 29 of 4.4.2.6.5.1.                               |
| macListMPRegDevices | 0x2050 | BN | List of registered devices. This list is maintained by the Base Node only. Each entry in this list shall comprise the following information. |                         |                                                                                                                                                                                       |
|                     |        |    | Entry Element                                                                                                                                | Type                    | Description                                                                                                                                                                           |
|                     |        |    | regEntryID                                                                                                                                   | EUI-48                  | EUI-48 of the registered Node.                                                                                                                                                        |
|                     |        |    | regEntryLNID                                                                                                                                 | Integer16               | LNID allocated to this Node.                                                                                                                                                          |
|                     |        |    | regEntryState                                                                                                                                | TERMINAL=1,<br>SWITCH=2 | Functional state of this Node.                                                                                                                                                        |
|                     |        |    | regEntryLSID                                                                                                                                 | Integer8                | SID allocated to this Node. This entry element corresponds to a PRO.NSID obtained in a promotion accomplished using the PRO_REQ_x messages.                                           |
|                     |        |    | regEntrySID                                                                                                                                  | Integer8                | SID of Switch through which this Node is connected.                                                                                                                                   |

| List Attribute Name | Id | M | Description    |           |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |
|---------------------|----|---|----------------|-----------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|                     |    |   | regEntryLevel  | Integer8  | Hierarchy level of this Node.                                                                                                                                                                                                                                                                                                                                                                                                                                                                           |
|                     |    |   | regEntryTCap   | Integer8  | <p>Bitmap of MAC Capabilities of Terminal functions in this device.</p> <p>Bits in sequence of right-to-left shall have the following meaning:</p> <p><b>Bit0:</b> Robust mode Capable;<br/> <b>Bit1:</b> Backward Compatible Capable;<br/> <b>Bit2:</b> Switch Capable;<br/> <b>Bit3:</b> Packet Aggregation Capable;<br/> <b>Bit4:</b> Connection Free Period Capable;<br/> <b>Bit5:</b> Direct Connection Capable;<br/> <b>Bit6:</b> ARQ Capable;<br/> <b>Bit7:</b> Multi-PHY Promotion Capable.</p> |
|                     |    |   | regEntrySwCap  | Integer8  | <p>Bitmap of MAC Switching capabilities of this device</p> <p>Bits in sequence of right-to-left shall have the following meaning:</p> <p><b>Bit0:</b> Direct Connection Switching;<br/> <b>Bit1:</b> Reserved;<br/> <b>Bit2:</b> Reserved;<br/> <b>Bit3:</b> ARQ Buffering Switching Capability;<br/> <b>Bit4 to 7:</b> Reserved for future use.</p>                                                                                                                                                    |
|                     |    |   | regEntryMPLSID | Integer8  | SID allocated to this Node. This entry element corresponds to a PRO.NSID obtained in a promotion accomplished using the PRO_REQ_x_MultiPHY messages.                                                                                                                                                                                                                                                                                                                                                    |
|                     |    |   | regEntryPCH    | Integer16 | The medium and the band (PLC) or channel (RF) used by the node for its registration. This field is coded as the PRO.PCH field of Table 29 of 4.4.2.6.5.1. This field may be also associated to regEntryLSID.                                                                                                                                                                                                                                                                                            |



| List Attribute Name        | Id     | M   | Description                                                                                                                                                                                                                                                 |                 |                                                                                                                                                  |
|----------------------------|--------|-----|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-----------------|--------------------------------------------------------------------------------------------------------------------------------------------------|
|                            |        |     | regEntryMPPCH                                                                                                                                                                                                                                               | Integer16       | The medium and the band (PLC) or channel (RF) associated to regEntryMPLSID. This field is coded as the PRO.PCH field of Table 29 of 4.4.2.6.5.1. |
| macListMPAvailableSwitches | 0x2056 | SN  | List of Switch Nodes whose beacons are received.                                                                                                                                                                                                            |                 |                                                                                                                                                  |
|                            |        |     | Entry Element                                                                                                                                                                                                                                               | Type            | Description                                                                                                                                      |
|                            |        |     | slistEntrySNA                                                                                                                                                                                                                                               | EUI-48          | EUI-48 of the Subnetwork.                                                                                                                        |
|                            |        |     | slistEntryLSID                                                                                                                                                                                                                                              | Integer8        | SID of this Switch.                                                                                                                              |
|                            |        |     | slistEntryLevel                                                                                                                                                                                                                                             | Integer8        | Level of this Switch in Subnetwork hierarchy.                                                                                                    |
|                            |        |     | slistEntryRxLvl                                                                                                                                                                                                                                             | Integer8<br>EMA | Received signal level for this Switch. EMA output is determined using values that correspond to the Level parameter defined in 3.5.2.4.2.        |
|                            |        |     | slistEntryRxSNR                                                                                                                                                                                                                                             | Integer8<br>EMA | Signal to Noise Ratio for this Switch. EMA output is determined using values that correspond to the SNR parameter defined in 3.5.3.12.2.         |
|                            |        |     | slistEntryPCH                                                                                                                                                                                                                                               | Integer16       | The medium and the band (PLC) or channel (RF) of the received beacon. This field is coded as the PRO.PCH field of Table 29 of 4.4.2.6.5.1.       |
| macListMPPhyComm           | 0x2059 | All | List of PHY communication parameters. This table is maintained in every Node. For Terminal Nodes it contains only one entry for the Switch the Node is connected through. For other Nodes it contains also entries for every directly connected child Node. |                 |                                                                                                                                                  |
|                            |        |     | Entry Element                                                                                                                                                                                                                                               | Type            | Description                                                                                                                                      |
|                            |        |     | phyCommLNID                                                                                                                                                                                                                                                 | Integer16       | LNID of the peer device                                                                                                                          |
|                            |        |     | phyCommSID                                                                                                                                                                                                                                                  | Integer8        | SID of the peer device                                                                                                                           |

| List Attribute Name | Id | M | Description              |                 |                                                                                                                                                                                               |
|---------------------|----|---|--------------------------|-----------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|                     |    |   | phyCommTxPwr             | Integer8        | Tx power of GPDU packets sent to the device.                                                                                                                                                  |
|                     |    |   | phyCommRxLvl             | Integer8<br>EMA | Rx power level of GPDU packets received from the device. EMA output is determined using values that correspond to the Level parameter defined in 3.5.2.4.2.                                   |
|                     |    |   | phyCommSNR               | Integer8<br>EMA | SNR of GPDU packets received from the device. EMA output is determined using values that correspond to the SNR parameter defined in 3.5.3.12.2.                                               |
|                     |    |   | phyCommTxModulation      | Integer8        | For PLC, modulation scheme to be used for communicating with this node.<br><br>For RF this field is reserved for future use.                                                                  |
|                     |    |   | phyCommPhyTypeCapability | Integer8        | For PLC, capability of the node to receive only PHY Type A or PHY Type A+B frames<br><br>0: Type A only node<br>1: Type A+B capable node<br><br>For RF this field is reserved for future use. |
|                     |    |   | phyCommRxAge             | Integer16       | For PLC, time [seconds] since last update of phyCommTxModulation.<br><br>For RF this field is reserved for future use.                                                                        |
|                     |    |   | phyCommPCH               | Integer16       | The medium and the band (PLC) or channel (RF) used to communicate with this node. This field is coded as the PRO.PCH field of Table 29 of 4.4.2.6.5.1.                                        |

### 6.2.3.6 MAC security attributes

Table 112 – MAC security attributes

| Attribute Name    | Id     | Size     | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             |
|-------------------|--------|----------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| macSecDUK         | 0x005B | 128 bits | <p>Device Unique Key to use in initial key derivation functions. The key shall be updated immediately; it shall not require re-registering the node.</p> <p>As a guideline, the Base Node should store both the old and new keys until the node has complete a successful registration with the new one, in the reception of a REG_REQ the Base Node should authenticate with the new key and if failed try again with the old one.</p> <p>Access to this PIB shall have the following restrictions:</p> <ul style="list-style-type: none"> <li>• Is write only, shall not be read.</li> <li>• Shall only be available if the underlying connection is encrypted, authenticated and unicast.</li> </ul> |
| MACUpdateKeysTime | 0x005C | 32 bits  | <p>Maximum time in seconds allowed using the same SWK or WK. After that time the keys shall no longer be considered valid.</p> <p>The maximum allowed value for this PIB is defined in Table 15, the value depends on the number of channels.</p>                                                                                                                                                                                                                                                                                                                                                                                                                                                       |

### 6.2.3.7 Action PIB attributes

Some of the conformance tests require triggering certain actions on Service Nodes and Base Nodes. The following table lists the set of action attributes that need to be supported by all implementations.

Table 113 - Action PIB attributes

| Attribute Name      | Id     | M  | Size (bytes) | Description                                                                    |
|---------------------|--------|----|--------------|--------------------------------------------------------------------------------|
| MACActionTxData     | 0x0060 | SN | 1            | Total number of PPDUs correctly decoded. Useful for PHY layer to estimate FER. |
| MACActionConnClose  | 0x0061 | SN | 1            | Trigger to close one of the open connections.                                  |
| MACActionRegReject  | 0x0062 | SN | 1            | Trigger to reject incoming registration request.                               |
| MACActionProReject  | 0x0063 | SN | 1            | Trigger to reject incoming promotion request.                                  |
| MACActionUnregister | 0x0064 | SN | 1            | Trigger to unregister from the Subnetwork.                                     |

| Attribute Name   | Id     | M     | Size (bytes) | Description                                                                                                                                                                                                                                                          |                  |                                                                                             |
|------------------|--------|-------|--------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|------------------|---------------------------------------------------------------------------------------------|
| MACActionPromote | 0x0065 | BN    | 6            | Trigger to promote a given Service Node from the Subnetwork. This action refers to a promotion process on the same medium and on the same band (PLC) or channel (RF) a Service Node is connected to the Subnetwork.<br><br>PARAM: EUI-48 of the node being promoted. |                  |                                                                                             |
| MACActionDemote  | 0x0066 | BN    | 6            | Trigger to demote a given Service Node from the Subnetwork. This action refers to a demotion process on the same medium and on the same band (PLC) or channel (RF) a Service Node is connected to the Subnetwork.<br><br>PARAM: EUI-48 of the node being demoted.    |                  |                                                                                             |
| MACActionReject  | 0x0067 | BN    |              | Rejects or stops (toggles) rejecting packets of a certain type                                                                                                                                                                                                       |                  |                                                                                             |
|                  |        |       |              | Entry Element                                                                                                                                                                                                                                                        | Size             | Description                                                                                 |
|                  |        |       |              | Node                                                                                                                                                                                                                                                                 | 6                | EUI 48 of the Node                                                                          |
|                  |        |       |              | Reject                                                                                                                                                                                                                                                               | 1                | 1 – Reject<br>0 – Stop Rejecting                                                            |
|                  |        |       |              | Type                                                                                                                                                                                                                                                                 | 1                | 0 – rejects PRO_REQ_S<br>1 – rejects PRM<br>2 – rejects CON_REQ_S<br><br>3- rejects REG_REQ |
| MACAliveTime     | 0x0068 | BN    | 1            | Forces alive time for the network or sets it as automatic.                                                                                                                                                                                                           |                  |                                                                                             |
|                  |        |       |              | Value                                                                                                                                                                                                                                                                | macAliveTimeMode |                                                                                             |
|                  |        |       |              |                                                                                                                                                                                                                                                                      | 0                | 1                                                                                           |
|                  |        |       |              | 0x00                                                                                                                                                                                                                                                                 | 128s             | 32s                                                                                         |
|                  |        |       |              | 0x01                                                                                                                                                                                                                                                                 | 256s             | 64s                                                                                         |
|                  |        |       |              | 0x02                                                                                                                                                                                                                                                                 | 512s             | 128s                                                                                        |
|                  |        |       |              | 0x03                                                                                                                                                                                                                                                                 | 2048s            | 256s                                                                                        |
|                  |        |       |              | 0x04                                                                                                                                                                                                                                                                 | 4096s            | 512s                                                                                        |
|                  |        |       |              | 0x05                                                                                                                                                                                                                                                                 | 8192s            | 1024s                                                                                       |
|                  |        |       |              | 0x06                                                                                                                                                                                                                                                                 | 16384s           | 2048s                                                                                       |
| 0x07             | 32768s | 4096s |              |                                                                                                                                                                                                                                                                      |                  |                                                                                             |

| Attribute Name              | Id     | M  | Size (bytes) | Description                                                       |                                           |                                                                                         |
|-----------------------------|--------|----|--------------|-------------------------------------------------------------------|-------------------------------------------|-----------------------------------------------------------------------------------------|
|                             |        |    |              | 0xFF                                                              | Reset alive time to the configured value. |                                                                                         |
|                             | 0x0069 |    |              | Deprecated since v1.3.6 and reserved for future use               |                                           |                                                                                         |
| MACActionBroadcastDataBurst | 0x006A | BN |              | Send a burst of data PDU-s with a test sequence using broadcast   |                                           |                                                                                         |
|                             |        |    |              | Entry Element                                                     | Size                                      | Description                                                                             |
|                             |        |    |              | Number                                                            | 4                                         | Number of PDUs to be sent                                                               |
|                             |        |    |              | DataLength                                                        | 1                                         | Size of data packet to send                                                             |
|                             |        |    |              | DutyCycle                                                         | 1                                         | Average duty cycle (%). It must be the average because of the randomness of the CSMA/CA |
|                             |        |    |              | LCID                                                              | 2                                         | LCID of the broadcast data to be sent                                                   |
|                             |        |    |              | Priority                                                          | 1                                         | Priority of data packets to be sent                                                     |
| MACActionMgmtCon            | 0x006B | BN |              | Forces establishment/close of the management connection           |                                           |                                                                                         |
|                             |        |    |              | Entry Element                                                     | Size                                      | Description                                                                             |
|                             |        |    |              | Node                                                              | 6                                         | EUI-48 of the Service Node                                                              |
|                             |        |    |              | Connect                                                           | 1                                         | 0 – Close the management connection<br>1 – Open the management connection               |
| MACActionMgmtMul            | 0x006C | BN |              | Forces establishment/close of the management multicast connection |                                           |                                                                                         |
|                             |        |    |              | Entry Element                                                     | Size                                      | Description                                                                             |
|                             |        |    |              | Node                                                              | 6                                         | EUI-48 of the Service Node                                                              |

| Attribute Name           | Id     | M  | Size (bytes) | Description                                                                                                                                                                                                                                                                         |      |                                                                                               |
|--------------------------|--------|----|--------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|------|-----------------------------------------------------------------------------------------------|
|                          |        |    |              | Join                                                                                                                                                                                                                                                                                | 1    | 0 – Leave the management multicast connection<br>1 – Join the management multicast connection |
| MACActionUnregister BN   | 0x006D | BN | 6            | Trigger to unregister a given Service Node from the Subnetwork.<br><br>PARAM: EUI-48 of the node being unregistered.                                                                                                                                                                |      |                                                                                               |
| MACActionConnCloseBN     | 0x006E | BN |              | Trigger to close an open connection.                                                                                                                                                                                                                                                |      |                                                                                               |
|                          |        |    |              | Entry element                                                                                                                                                                                                                                                                       | Size | Description                                                                                   |
|                          |        |    |              | node                                                                                                                                                                                                                                                                                | 6    | Eui48 of the node                                                                             |
|                          |        |    |              | LCID                                                                                                                                                                                                                                                                                | 2    | LCID of the connection to be closed.                                                          |
| MACActionSegmented 4-32  | 0x006F | BN |              | <ul style="list-style-type: none"><li>• Trigger data transfer whit segmentation mechanism working (Convergence Layer)</li><li>• Transmit PPDUs over established CL 4-32 Connection (with segmentation)</li><li>• Trigger at least 1 packet segmented in at least 3 frames</li></ul> |      |                                                                                               |
|                          |        |    |              | Entry Element                                                                                                                                                                                                                                                                       | Size | Description                                                                                   |
|                          |        |    |              | Node                                                                                                                                                                                                                                                                                | 6    | EUI-48 of the Service Node                                                                    |
|                          |        |    |              | Length                                                                                                                                                                                                                                                                              | 2    | Length of the data being transmitted (number or segments will depend on this length)          |
| MACActionAppemuDataBurst | 0x0080 | BN |              | Send a burst of data PDU-s with a test sequence using the Appemu connection to the node (if any)<br>The data shall be transmitted with the flush bit to zero (0) when possible.                                                                                                     |      |                                                                                               |
|                          |        |    |              | Entry Element                                                                                                                                                                                                                                                                       | Size | Description                                                                                   |
|                          |        |    |              | Node                                                                                                                                                                                                                                                                                | 6    | EUI-48 of the Service Node                                                                    |
|                          |        |    |              | Number                                                                                                                                                                                                                                                                              | 4    | Number of PDU-s to be sent                                                                    |

| Attribute Name          | Id     | M | Size (bytes) | Description                                                                                                                                                                           |      |                                                                                                  |
|-------------------------|--------|---|--------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|------|--------------------------------------------------------------------------------------------------|
|                         |        |   |              | DataLength                                                                                                                                                                            | 1    | Size of the data packets to be sent                                                              |
|                         |        |   |              | DutyCycle                                                                                                                                                                             | 1    | Average duty cycle (percentage). It must be the average because of the randomness of the CSMA/CA |
| MACActionMgmtData Burst | 0x0081 | M |              | Send a burst of data PDU-s with a test sequence using the Management connection to the node (if any). The data shall be transmitted with the flush bit set to zero (0) when possible. |      |                                                                                                  |
|                         |        |   |              | Entry Element                                                                                                                                                                         | Size | Description                                                                                      |
|                         |        |   |              | Node                                                                                                                                                                                  | 6    | EUI-48 of the Service Node, In service node this field will be ignored                           |
|                         |        |   |              | Number                                                                                                                                                                                | 4    | Number of PDU-s to be sent                                                                       |
|                         |        |   |              | DataLength                                                                                                                                                                            | 1    | Size of the data packets to be sent                                                              |
|                         |        |   |              | DutyCycle                                                                                                                                                                             | 1    | Average duty cycle (percentage). It must be the average because of the randomness of the CSMA/CA |

#### 5261 6.2.3.8 MAC Network performance attributes

| Attribute Name              | Id     | M  | Size (bytes) | Valid Range    | Description                                                                                                                                                                                                                                                              |
|-----------------------------|--------|----|--------------|----------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| macNetworkUpTime            | 0x0100 | BN | 4            | 0 – 0xFFFF FFF | Period in seconds from the instant when the BN started to send beacons. This attribute is maintained only in the Base Node.                                                                                                                                              |
| macNetworkAllocated Beacons | 0x0101 | BN | 2            | 0 - 16256      | Allocated PLC beacons in a superframe, calculated as the sum of all PLC beacons transmitted in the Subnetwork during a superframe. The number of PLC beacons transmitted by a device in a superframe depends on the transmission frequency (BCN.FRQ) associated to BPDU. |

| Attribute Name                   | Id     | M  | Size (bytes) | Valid Range   | Description                                                                                                                                                                                             |
|----------------------------------|--------|----|--------------|---------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|                                  |        |    |              |               | This attribute is maintained only in the Base Node.                                                                                                                                                     |
| macNetworkPromotionsCounter      | 0x0102 | BN | 4            | 0 – 0xFFFFFFF | <p>Number of successful Promotions processes in Service Nodes connected through PLC medium (single and double) since last network reset.</p> <p>This attribute is maintained only in the Base Node.</p> |
| macNetworkDemotionsCounter       | 0x0103 | BN | 4            | 0 – 0xFFFFFFF | <p>Number of Demotions events in Service Nodes connected through PLC medium (single and double) since last network reset.</p> <p>This attribute is maintained only in the Base Node.</p>                |
| macNetworkRegistrationsCounter   | 0x0104 | BN | 4            | 0 – 0xFFFFFFF | <p>Number of successful Registration processes in Service Nodes connected through PLC medium since last network reset.</p> <p>This attribute is maintained only in the Base Node.</p>                   |
| macNetworkUnregistrationsCounter | 0x0105 | BN | 4            | 0 – 0xFFFFFFF | <p>Number of Unregistration events in Service Nodes connected through PLC medium since last network reset.</p> <p>This attribute is maintained only in the Base Node.</p>                               |
| macNetworkCoverage               | 0x0106 | BN | 2            | 0-10000       | <p>Network coverage value expressed in percentage hundredths. See Annex P to see how to calculate it.</p> <p>This attribute is maintained only in the Base Node.</p>                                    |
| macNetworkNodesCount             | 0x0107 | BN | 2            | 0-65535       | Number of unique Service Nodes that completed registration process with success at least once since last network reset.                                                                                 |



| Attribute Name         | Id           | M                       | Size (bytes) | Valid Range | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             |               |              |             |                        |   |                         |
|------------------------|--------------|-------------------------|--------------|-------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|---------------|--------------|-------------|------------------------|---|-------------------------|
|                        |              |                         |              |             | This attribute is maintained only in the Base Node.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |               |              |             |                        |   |                         |
| macNetworkAvailability | 0x0108       | BN                      | 6            |             | <ul style="list-style-type: none"><li>Time, in seconds, in which Service Nodes were in registered state, calculated as follow:<br/><math display="block">\sum_{i=1}^n (Treg_i)</math><br/>Where:<br/><br/>Treg: is the time in which a particular node was considered in registered state since the previous reading of the attribute. The Treg time can be referred to more than a single time interval, e.g. if an unregistration and a following new registration events associated to the same node occur both after the previous reading of the attribute and before the current reading, the Treg value associated to node will be the sum of 2 different time intervals.<br/><br/>n: is the number of unique nodes that were considered in registered state at least once since the previous reading of the attribute. The number is not decreased in case an unregistration event occurs after the previous reading of the attribute and before the current reading.</li><li>Number of nodes involved in previous calculation (n).</li></ul> <p>This attribute is reset after each reading.</p> <p>This attribute is maintained only in the Base Node.</p> <table><tr><th>Entry Element</th><th>Size (bytes)</th><th>Description</th></tr><tr><td>LastPeriodAvailability</td><td>4</td><td><math display="block">\sum_{i=1}^n (Treg_i)</math></td></tr></table> | Entry Element | Size (bytes) | Description | LastPeriodAvailability | 4 | $\sum_{i=1}^n (Treg_i)$ |
| Entry Element          | Size (bytes) | Description             |              |             |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |               |              |             |                        |   |                         |
| LastPeriodAvailability | 4            | $\sum_{i=1}^n (Treg_i)$ |              |             |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |               |              |             |                        |   |                         |

| Attribute Name                     | Id     | M  | Size (bytes) | Valid Range   | Description                                                                                                                                                                                                                                                           |   |   |  |
|------------------------------------|--------|----|--------------|---------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|---|---|--|
|                                    |        |    |              |               | NodesNum                                                                                                                                                                                                                                                              | 2 | N |  |
| macRFNetworkAllocatedBeacons       | 0x0109 | BN | 2            | 0 - 16256     | Allocated RF beacons in a superframe, calculated as the sum of all RF beacons transmitted in the subnetwork during a superframe. The number of RF beacons transmitted by a device in a superframe depends on the transmission frequency (BCN.FRQ) associated to BPDU. |   |   |  |
| macRFNetworkPromotionsCounter      | 0x010A | BN | 4            | 0 – 0xFFFFFFF | Number of successful Promotions processes in Service Nodes connected through RF medium since last network reset.                                                                                                                                                      |   |   |  |
| macRFNetworkDemotionsCounter       | 0x010B | BN | 4            | 0 – 0xFFFFFFF | Number of Demotions events in Service Nodes connected through RF medium since last network reset.                                                                                                                                                                     |   |   |  |
| macRFNetworkRegistrationsCounter   | 0x010C | BN | 4            | 0 – 0xFFFFFFF | Number of successful Registration processes in Service Nodes connected through RF medium since last network reset.                                                                                                                                                    |   |   |  |
| macRFNetworkUnregistrationsCounter | 0x010D | BN | 4            | 0 – 0xFFFFFFF | Number of Unregistration events in Service Nodes connected through RF medium since last network reset.                                                                                                                                                                |   |   |  |

| macListExtDevicesInfo | 0x0150    | <p>List of Service Nodes registered successfully at least once since last reset of Base Node. This list is maintained by the Base Node only. Each entry in this list shall comprise the following information:</p> <table border="1"> <thead> <tr> <th>Entry Element</th><th>Type</th><th>Description</th></tr> </thead> <tbody> <tr> <td>EntryID</td><td>EUI48</td><td>EUI48 of the registered node</td></tr> <tr> <td>EntryLNID</td><td>Integer16</td><td>LNID allocated to this node (0xFFFF if device is in Disconnected state)</td></tr> <tr> <td>EntryState</td><td>Integer8</td><td>Functional state of this node (DISCONNECTED=0, TERMINAL=1, SWITCH=2)</td></tr> <tr> <td>EntrySID</td><td>Integer8</td><td>SID of Switch through which this node is connected (0xFF if device is in Disconnected).</td></tr> <tr> <td>EntryLSID</td><td>Integer8</td><td>SID allocated to this Node (0xFF if device is in Disconnected or Terminal state). This entry element is associated to a promotion accomplished using the PRO_REQ_x messages</td></tr> <tr> <td>EntryPCH</td><td>Integer16</td><td>The medium and the band (PLC) or channel (RF) used by the node for registration. This field is coded as the PRO.PCH field of Table 28 of 4.4.2.6.5.1 (0xFFFF if device is in Disconnected state)</td></tr> <tr> <td>EntryMPLSID</td><td>Integer8</td><td>SID allocated to this Node (0xFF if device is in Disconnected or Terminal state). This entry element is associated to a promotion accomplished using the PRO_REQ_x_MultiPHY message</td></tr> <tr> <td>EntryMPPCH</td><td>Integer16</td><td>Codification of the physical layer associated to EntryMPLSID indicating to where the switch is promoted. This field is coded as the PRO.PCH field of Table 28 of 4.4.2.6.5.1 (0xFFFF if device is in Disconnected state or not a Switch in medium different from one registered through)</td></tr> <tr> <td>EntryLevel</td><td>Integer8</td><td>Hierarchy level of this node (0 if device is in Disconnected state)</td></tr> <tr> <td>EntryTCap</td><td>Integer8</td><td> <p>Bitmap of MAC Capabilities of Terminal functions in this device</p> <p>Bits in sequence of right-to-left shall have the following meaning:</p> <p><b>Bit0:</b> Robust mode Capable;</p> </td></tr> </tbody> </table> | Entry Element | Type | Description | EntryID | EUI48 | EUI48 of the registered node | EntryLNID | Integer16 | LNID allocated to this node (0xFFFF if device is in Disconnected state) | EntryState | Integer8 | Functional state of this node (DISCONNECTED=0, TERMINAL=1, SWITCH=2) | EntrySID | Integer8 | SID of Switch through which this node is connected (0xFF if device is in Disconnected). | EntryLSID | Integer8 | SID allocated to this Node (0xFF if device is in Disconnected or Terminal state). This entry element is associated to a promotion accomplished using the PRO_REQ_x messages | EntryPCH | Integer16 | The medium and the band (PLC) or channel (RF) used by the node for registration. This field is coded as the PRO.PCH field of Table 28 of 4.4.2.6.5.1 (0xFFFF if device is in Disconnected state) | EntryMPLSID | Integer8 | SID allocated to this Node (0xFF if device is in Disconnected or Terminal state). This entry element is associated to a promotion accomplished using the PRO_REQ_x_MultiPHY message | EntryMPPCH | Integer16 | Codification of the physical layer associated to EntryMPLSID indicating to where the switch is promoted. This field is coded as the PRO.PCH field of Table 28 of 4.4.2.6.5.1 (0xFFFF if device is in Disconnected state or not a Switch in medium different from one registered through) | EntryLevel | Integer8 | Hierarchy level of this node (0 if device is in Disconnected state) | EntryTCap | Integer8 | <p>Bitmap of MAC Capabilities of Terminal functions in this device</p> <p>Bits in sequence of right-to-left shall have the following meaning:</p> <p><b>Bit0:</b> Robust mode Capable;</p> |
|-----------------------|-----------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|---------------|------|-------------|---------|-------|------------------------------|-----------|-----------|-------------------------------------------------------------------------|------------|----------|----------------------------------------------------------------------|----------|----------|-----------------------------------------------------------------------------------------|-----------|----------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------|----------|-----------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-------------|----------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|------------|-----------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|------------|----------|---------------------------------------------------------------------|-----------|----------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Entry Element         | Type      | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |               |      |             |         |       |                              |           |           |                                                                         |            |          |                                                                      |          |          |                                                                                         |           |          |                                                                                                                                                                             |          |           |                                                                                                                                                                                                  |             |          |                                                                                                                                                                                     |            |           |                                                                                                                                                                                                                                                                                          |            |          |                                                                     |           |          |                                                                                                                                                                                            |
| EntryID               | EUI48     | EUI48 of the registered node                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 |               |      |             |         |       |                              |           |           |                                                                         |            |          |                                                                      |          |          |                                                                                         |           |          |                                                                                                                                                                             |          |           |                                                                                                                                                                                                  |             |          |                                                                                                                                                                                     |            |           |                                                                                                                                                                                                                                                                                          |            |          |                                                                     |           |          |                                                                                                                                                                                            |
| EntryLNID             | Integer16 | LNID allocated to this node (0xFFFF if device is in Disconnected state)                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |               |      |             |         |       |                              |           |           |                                                                         |            |          |                                                                      |          |          |                                                                                         |           |          |                                                                                                                                                                             |          |           |                                                                                                                                                                                                  |             |          |                                                                                                                                                                                     |            |           |                                                                                                                                                                                                                                                                                          |            |          |                                                                     |           |          |                                                                                                                                                                                            |
| EntryState            | Integer8  | Functional state of this node (DISCONNECTED=0, TERMINAL=1, SWITCH=2)                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |               |      |             |         |       |                              |           |           |                                                                         |            |          |                                                                      |          |          |                                                                                         |           |          |                                                                                                                                                                             |          |           |                                                                                                                                                                                                  |             |          |                                                                                                                                                                                     |            |           |                                                                                                                                                                                                                                                                                          |            |          |                                                                     |           |          |                                                                                                                                                                                            |
| EntrySID              | Integer8  | SID of Switch through which this node is connected (0xFF if device is in Disconnected).                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |               |      |             |         |       |                              |           |           |                                                                         |            |          |                                                                      |          |          |                                                                                         |           |          |                                                                                                                                                                             |          |           |                                                                                                                                                                                                  |             |          |                                                                                                                                                                                     |            |           |                                                                                                                                                                                                                                                                                          |            |          |                                                                     |           |          |                                                                                                                                                                                            |
| EntryLSID             | Integer8  | SID allocated to this Node (0xFF if device is in Disconnected or Terminal state). This entry element is associated to a promotion accomplished using the PRO_REQ_x messages                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |               |      |             |         |       |                              |           |           |                                                                         |            |          |                                                                      |          |          |                                                                                         |           |          |                                                                                                                                                                             |          |           |                                                                                                                                                                                                  |             |          |                                                                                                                                                                                     |            |           |                                                                                                                                                                                                                                                                                          |            |          |                                                                     |           |          |                                                                                                                                                                                            |
| EntryPCH              | Integer16 | The medium and the band (PLC) or channel (RF) used by the node for registration. This field is coded as the PRO.PCH field of Table 28 of 4.4.2.6.5.1 (0xFFFF if device is in Disconnected state)                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             |               |      |             |         |       |                              |           |           |                                                                         |            |          |                                                                      |          |          |                                                                                         |           |          |                                                                                                                                                                             |          |           |                                                                                                                                                                                                  |             |          |                                                                                                                                                                                     |            |           |                                                                                                                                                                                                                                                                                          |            |          |                                                                     |           |          |                                                                                                                                                                                            |
| EntryMPLSID           | Integer8  | SID allocated to this Node (0xFF if device is in Disconnected or Terminal state). This entry element is associated to a promotion accomplished using the PRO_REQ_x_MultiPHY message                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |               |      |             |         |       |                              |           |           |                                                                         |            |          |                                                                      |          |          |                                                                                         |           |          |                                                                                                                                                                             |          |           |                                                                                                                                                                                                  |             |          |                                                                                                                                                                                     |            |           |                                                                                                                                                                                                                                                                                          |            |          |                                                                     |           |          |                                                                                                                                                                                            |
| EntryMPPCH            | Integer16 | Codification of the physical layer associated to EntryMPLSID indicating to where the switch is promoted. This field is coded as the PRO.PCH field of Table 28 of 4.4.2.6.5.1 (0xFFFF if device is in Disconnected state or not a Switch in medium different from one registered through)                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |               |      |             |         |       |                              |           |           |                                                                         |            |          |                                                                      |          |          |                                                                                         |           |          |                                                                                                                                                                             |          |           |                                                                                                                                                                                                  |             |          |                                                                                                                                                                                     |            |           |                                                                                                                                                                                                                                                                                          |            |          |                                                                     |           |          |                                                                                                                                                                                            |
| EntryLevel            | Integer8  | Hierarchy level of this node (0 if device is in Disconnected state)                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |               |      |             |         |       |                              |           |           |                                                                         |            |          |                                                                      |          |          |                                                                                         |           |          |                                                                                                                                                                             |          |           |                                                                                                                                                                                                  |             |          |                                                                                                                                                                                     |            |           |                                                                                                                                                                                                                                                                                          |            |          |                                                                     |           |          |                                                                                                                                                                                            |
| EntryTCap             | Integer8  | <p>Bitmap of MAC Capabilities of Terminal functions in this device</p> <p>Bits in sequence of right-to-left shall have the following meaning:</p> <p><b>Bit0:</b> Robust mode Capable;</p>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |               |      |             |         |       |                              |           |           |                                                                         |            |          |                                                                      |          |          |                                                                                         |           |          |                                                                                                                                                                             |          |           |                                                                                                                                                                                                  |             |          |                                                                                                                                                                                     |            |           |                                                                                                                                                                                                                                                                                          |            |          |                                                                     |           |          |                                                                                                                                                                                            |

|  |  |                        |           |                                                                                                                                                                                                                                                                                                                                        |
|--|--|------------------------|-----------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|  |  |                        |           | <b>Bit1:</b> Backward Compatible Capable;<br><b>Bit2:</b> Switch Capable;<br><b>Bit3:</b> Packet Aggregation Capable;<br><b>Bit4:</b> Connection Free Period Capable;<br><b>Bit5:</b> Direct Connection Capable;<br><b>Bit6:</b> ARQ Capable;<br><b>Bit7:</b> Multi-PHY Promotion Capable.                                             |
|  |  | EntrySwCap             | Integer8  | Bitmap of MAC Switching capabilities of this device.<br><br>Bits in sequence of right-to-left shall have the following meaning:<br><br><b>Bit0:</b> Direct Connection Switching;<br><b>Bit1:</b> Reserved;<br><b>Bit2:</b> Reserved;<br><b>Bit3:</b> ARQ Buffering Switching Capability;<br><b>Bit4 to 7:</b> Reserved for future use. |
|  |  | EntryPLCAvailability   | Integer32 | Total Time, in seconds, in which the Service Node is in registered state and connected through PLC medium                                                                                                                                                                                                                              |
|  |  | EntryRFAvailability    | Integer32 | Total Time, in seconds, in which the Service Node is in registered state and connected through RF medium                                                                                                                                                                                                                               |
|  |  | EntryPLCDisconnections | Integer32 | Total number of Unregistration events from PLC medium associated to device                                                                                                                                                                                                                                                             |
|  |  | EntryRFDisconnections  | Integer32 | Total number of Unregistration events from RF medium associated to device                                                                                                                                                                                                                                                              |
|  |  | EntryPLCPromotions     | Integer32 | Total number of successful Promotions processes (single and double) in PLC medium associated to device                                                                                                                                                                                                                                 |
|  |  | EntryRFPromotions      | Integer32 | Total number of successful Promotions processes in RF medium associated to device                                                                                                                                                                                                                                                      |
|  |  | EntryAlvTime           | Integer8  | Current Alive time (0 if device is in Disconnected state)                                                                                                                                                                                                                                                                              |

|  |  |               |           |                                                                                                                |
|--|--|---------------|-----------|----------------------------------------------------------------------------------------------------------------|
|  |  | EntryCoverage | Integer16 | Device coverage in network, value expressed in percentage hundredths. See Annex XX to see how to calculate it. |
|--|--|---------------|-----------|----------------------------------------------------------------------------------------------------------------|

5262 **6.2.4 Application PIB attributes**

5263 The following PIB attributes are used for general administration and maintenance of a OFDM PRIME  
5264 compliant device. These attributes do not affect the communication functionality, but enable easier  
5265 administration.

5266 These attributes shall be supported by both Base Node and Service Node devices.

5267 **Table 114 - Applications PIB attributes**

| Attribute Name | Size (in bits) | Id     | Description                                                |
|----------------|----------------|--------|------------------------------------------------------------|
| AppFwVersion   | 128            | 0x0075 | Textual description of firmware version running on device. |
| AppVendorId    | 16             | 0x0076 | PRIME Alliance assigned unique vendor identifier.          |
| AppProductId   | 16             | 0x0077 | Vendor assigned unique identifier for specific product.    |

| Attribute Name                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        | Size<br>(in bits) | Id                                                                                                                                                                                                                                                                                                                                                                                                                                                     | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           |               |             |             |          |      |                                                                                                                                                                                                                                                                                                                                                                                                                                                        |
|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|---------------|-------------|-------------|----------|------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| AppListZCStatus                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |                   | 0x0078                                                                                                                                                                                                                                                                                                                                                                                                                                                 | Zero Cross Status list. This list contains entry for each available zero cross detection circuits available in the system. Each element is sent together with reference time of zero cross close to frame beginning. If multiple entries are requested at the same time only first will be replied.                                                                                                                                                                                                                                                                                                   |               |             |             |          |      |                                                                                                                                                                                                                                                                                                                                                                                                                                                        |
|                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |                   |                                                                                                                                                                                                                                                                                                                                                                                                                                                        | <table><tr><th>Entry element</th><th>Type</th><th>Description</th></tr><tr><td>ZCStatus</td><td>Byte</td><td><b>Bit 7 : reserved</b>, always 0<br/><b>Bits 5-6 : Terminal Block number</b><br/>0 : invalid<br/>1 : terminal block 1<br/>2 : terminal block 2<br/>3 : terminal block 3<br/><b>Bits 3-4 : Direction</b><br/>0 : unknown direction<br/>1 : falling<br/>2 : raising<br/>3 : reserved<br/><b>Bits 0-2: Status</b><br/>0 : available but unknown status<br/>1 : regular at 50Hz<br/>2 : regular at 60Hz<br/>3-5: reserved<br/>6: irregular intervals<br/>7: not available</td></tr></table> | Entry element | Type        | Description | ZCStatus | Byte | <b>Bit 7 : reserved</b> , always 0<br><b>Bits 5-6 : Terminal Block number</b><br>0 : invalid<br>1 : terminal block 1<br>2 : terminal block 2<br>3 : terminal block 3<br><b>Bits 3-4 : Direction</b><br>0 : unknown direction<br>1 : falling<br>2 : raising<br>3 : reserved<br><b>Bits 0-2: Status</b><br>0 : available but unknown status<br>1 : regular at 50Hz<br>2 : regular at 60Hz<br>3-5: reserved<br>6: irregular intervals<br>7: not available |
|                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |                   |                                                                                                                                                                                                                                                                                                                                                                                                                                                        | Entry element                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         | Type          | Description |             |          |      |                                                                                                                                                                                                                                                                                                                                                                                                                                                        |
| ZCStatus                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              | Byte              | <b>Bit 7 : reserved</b> , always 0<br><b>Bits 5-6 : Terminal Block number</b><br>0 : invalid<br>1 : terminal block 1<br>2 : terminal block 2<br>3 : terminal block 3<br><b>Bits 3-4 : Direction</b><br>0 : unknown direction<br>1 : falling<br>2 : raising<br>3 : reserved<br><b>Bits 0-2: Status</b><br>0 : available but unknown status<br>1 : regular at 50Hz<br>2 : regular at 60Hz<br>3-5: reserved<br>6: irregular intervals<br>7: not available |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |               |             |             |          |      |                                                                                                                                                                                                                                                                                                                                                                                                                                                        |
| <p>The description of the Status bits (0-2):</p> <ul style="list-style-type: none"><li>Available but unknown status: Zero cross time available, device is only capable to record the times, no internal validation done.</li><li>Regular at 50Hz: Zero cross time is available and regularly received with intervals that correspond to 50Hz.</li><li>Regular at 60Hz: Zero cross time is available and regularly received with intervals that correspond to 60Hz.</li><li>Irregular intervals: Zero cross time is available but intervals are irregular and do not correspond to 50Hz nor 60Hz.</li><li>Not available: Device does not receive zero cross.</li></ul> |                   |                                                                                                                                                                                                                                                                                                                                                                                                                                                        |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |               |             |             |          |      |                                                                                                                                                                                                                                                                                                                                                                                                                                                        |

## 5269 **6.3 Firmware upgrade**

### 5270 **6.3.1 General**

5271 The present section specifies firmware upgrade. Devices supporting PRIME may have several firmware inside  
5272 them, at least one supporting the Application itself, and the one related to the PRIME protocol. Although it  
5273 is possible that the application can perform the firmware upgrade of all the firmware images of the device,  
5274 for instance DLMS/COSEM image transfer, using COSEM image transfer object, supporting PRIME firmware  
5275 upgrade is mandatory in order to process to PRIME firmware upgrade independently of the application.

### 5276 **6.3.2 Requirements and features**

5277 This section specifies the firmware upgrade application, which is unique and mandatory for Base Nodes and  
5278 Service Nodes.

5279 The most important features of the Firmware Upgrade mechanism are listed below. See following chapters  
5280 for more information. The FU mechanism:

- 5281 • Shall be a part of management plane and therefore use the NULL SSSS, as specified in section 0
- 5282 • Is able to work in unicast (default mode) and multicast (optional mode). The control messages  
5283 are always sent using unicast connections, whereas data can be transmitted using both unicast  
5284 and multicast. No broadcast should be used to transmit data.
- 5285 • May change the data packet sizes according to the channel conditions. The packet size will not  
5286 be changed during the download process.
- 5287 • Is able to request basic information to the Service Nodes at anytime, such as device model,  
5288 firmware version and FU protocol version.
- 5289 • Shall be abortable at anytime.
- 5290 • Shall check the integrity of the downloaded FW after completing the reception. In case of failure,  
5291 the firmware upgrade application shall request a new retransmission.
- 5292 • The new firmware shall be executed in the Service Nodes only if they are commanded to do so.  
5293 The FU application shall have to be able to set the moment when the reset takes place.
- 5294 • Must be able to reject the new firmware after a “test” period and switch to the old version. The  
5295 duration of this test period has to be fixed by the FU mechanism.

### 5296 **6.3.3 General Description**

#### 5297 **6.3.3.1 General**

5298 The Firmware Upgrade mechanism is able to work in unicast and multicast modes. All control messages are  
5299 sent using unicast connections, whereas the data can be sent via unicast (by default) or multicast (only if  
5300 supported by the manufacturer). Note that in order to ensure correct reception of the FW when Service  
5301 Nodes from different vendors are upgraded, data packets shall not be sent via broadcast. Only unicast and  
5302 multicast are allowed. A Node will reply only to messages sent via unicast. See chapter 0 for a detailed  
5303 description of the control and information messages used by the FU mechanism.

5304 The unicast and multicast connections are set up by the Base Node. In case of supporting multicast, the Base  
5305 Node shall request the Nodes from a specific vendor to join a specific multicast group, which is exclusively  
5306 created to perform the firmware upgrade and is removed after finishing it.

5307 As said before, it is up to the vendor to use unicast or multicast for transmitting the data. In case of unicast  
5308 data transmission, please note that the use of ARQ is an optional feature. Some examples showing the traffic  
5309 between the Base Node and the Service Nodes in unicast and multicast are provided in 6.3.5.4.

5310 After completing the firmware download, each Service Node is committed by the Base Node to perform an  
5311 integrity check on it. The firmware download will be restarted if the firmware image results to be corrupt. In  
5312 other case, the Service Nodes will wait until they are commanded by the Base Node to execute the new  
5313 firmware.

5314 The FU mechanism can setup the instant when the recently downloaded firmware is executed on the Service  
5315 Nodes. Thus, the Base Node can choose to restart all Nodes at the same time or in several steps. After restart,  
5316 each Service Node runs the new firmware for a time period specified by the FU mechanism. If this period  
5317 expires without receiving any confirmation from the Base Node, or the Base Node decides to abort the  
5318 upgrade process, the Service Nodes will reject the new firmware and switch to the old version. In any other  
5319 case (a confirmation message is received) the Service Nodes will consider the new firmware as the only valid  
5320 version and delete the old one.

5321 This is done in order to leave an “open back-door” in case that the new firmware is defect or corrupt. Please  
5322 note that the Service Nodes are not allowed to discard any of the stored firmware versions until the final  
5323 confirmation from the Base Node arrives or until the safety time period expires. The two last firmware  
5324 upgrade steps explained above are shown in 0. See chapter 6.3.5.3 for a detailed description of the control  
5325 messages.



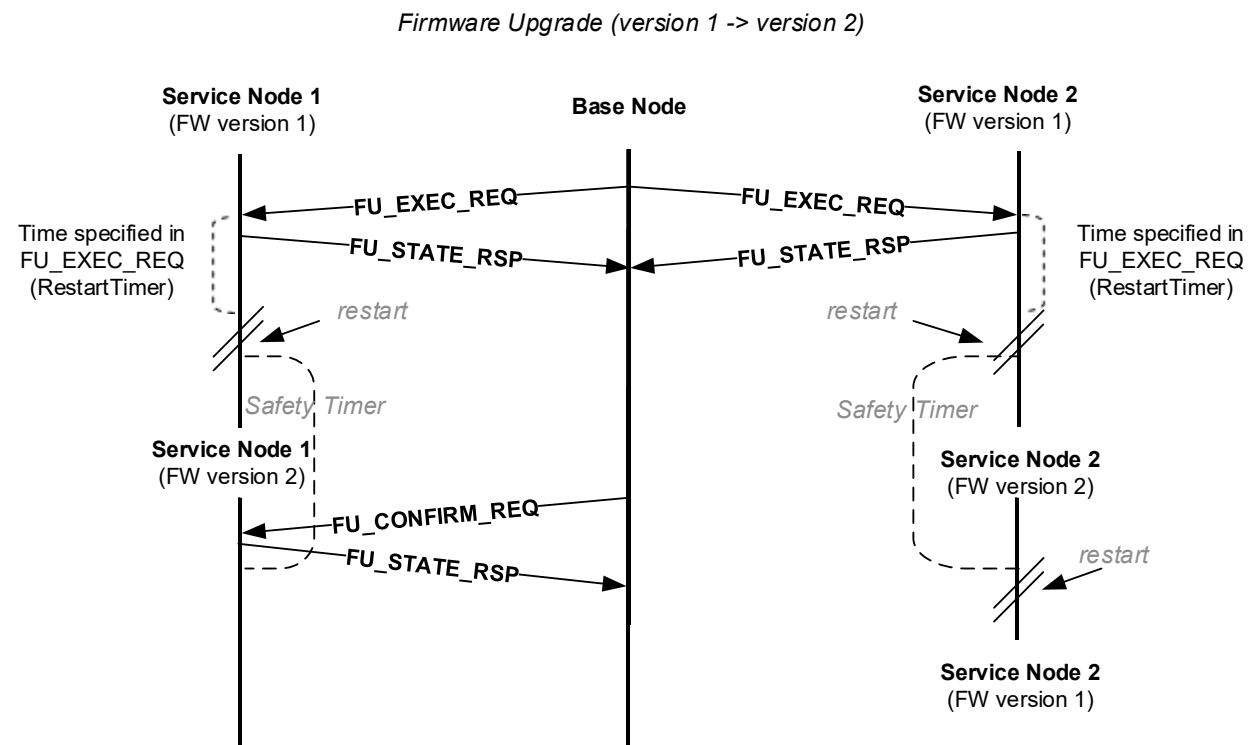


Figure 139 – Restarting de nodes and running the new firmware

**Note:** In normal circumstances, both Service Nodes should either accept or reject the new firmware version. Both possibilities are shown above simultaneously for academic purposes.

6.3.3.2 Signed firmware

The "signed firmware" refers to the concatenation of the Firmware Image and the signature as shown in the Figure 140. For now on, in the document will be refered as signed firmware.



Figure 140 – Signed firmware diagram

The payload transmitted in the Firmware Upgrade process shall be the signed firmware. For the SN to be able to differentiate both, the signature will have a length defined in the **FU\_INIT\_REQ**'s "Signature length" field.

6.3.3.3 Segmentation

The signed firmware is the information to be transferred, in order to process a firmware upgrade. The size of the signed firmware will be called "*ImageSize*", and is measured in bytes. This image is divided in smaller elements called pages that are easier to be transferred in packets. The "*PageSize*" may be one of the following: 32 bytes, 64 bytes, 128 bytes or 192 bytes. This implies that the number of pages in a signed firmware is calculated by the following formula:

5343 
$$PageCount = \left\lceil \frac{ImageSize}{PageSize} \right\rceil$$

5344 Every page will have a size specified by *PageSize*, except the last one that will contain the remaining bytes up  
5345 to *ImageSize*.

5346 The *PageSize* is configured by the Base Node and notified during the initialization of the Firmware Upgrade  
5347 process, and imposes a condition in the size of the packets being transferred by the protocol. The Service  
5348 Node must be able to handle all *PageSize* options to ensure proper receiving of the segmented data packets.

### 5349 6.3.4 Firmware upgrade PIB attributes

5350 The following PIB attributes shall be supported by Service Nodes to support the firmware download  
5351 application.

5352 Table 115 - FU PIB attributes

| Attribute Name    | Size<br>(in bits) | Id     | Description                                                                                                                       |
|-------------------|-------------------|--------|-----------------------------------------------------------------------------------------------------------------------------------|
| AppFwdlRunning    | 16                | 0x0070 | Indicate if a firmware download is in progress or not.<br><br>0 = No firmware download;<br><br>1 = Firmware download in progress. |
| AppFwdlRxPktCount | 16                | 0x0071 | Count of firmware download packets that have been received until the time of query.                                               |

5353

### 5354 6.3.5 State machine

#### 5355 6.3.5.1 General

5356 A Service Node using the Firmware Upgrade service will be in one of five possible states: *Idle*, *Receiving*,  
5357 *Complete*, *Countdown* and *Upgrade*. These states, the events triggering them and the resulting  
5358 actions/output messages are detailed below.

5359

5360 Table 116 - FU State Machine

|             | Description                          | Event                | Output (or action to be performed) | Next state       |
|-------------|--------------------------------------|----------------------|------------------------------------|------------------|
| <i>Idle</i> | The FU application is doing nothing. | Receive FU_INFO_REQ  | FU_INFO_RSP                        | <i>Idle</i>      |
|             |                                      | Receive FU_STATE_REQ | FU_STATE_RSP (.State = 0)          | <i>Idle</i>      |
|             |                                      | Receive FU_MISS_REQ  | FU_STATE_RSP (.State = 0)          | <i>Idle</i>      |
|             |                                      | Receive FU_INIT_REQ  | FU_STATE_RSP (.State = 1)          | <i>Receiving</i> |
|             |                                      | Receive FU_DATA      | (ignore)                           | <i>Idle</i>      |
|             |                                      | Receive FU_EXEC_REQ  | FU_STATE_RSP (.State = 0)          | <i>Idle</i>      |

|                  | Description                                                                                 | Event                                                   | Output (or action to be performed)          | Next state |
|------------------|---------------------------------------------------------------------------------------------|---------------------------------------------------------|---------------------------------------------|------------|
|                  |                                                                                             | Receive FU_CONFIRM_REQ                                  | FU_STATE_RSP (.State = 0)                   | Idle       |
|                  |                                                                                             | Receive FU_KILL_REQ                                     | FU_STATE_RSP (.State = 0)                   | Idle       |
|                  |                                                                                             | Any exception                                           |                                             | Exception  |
| <b>Receiving</b> | The FU application is receiving the Signed firmware.                                        | Complete FW received, CRC OK and Signature OK           |                                             | Complete   |
|                  |                                                                                             | Complete FW received and CRC not Ok or signature not OK |                                             | Exception  |
|                  |                                                                                             | Receive FU_INFO_REQ                                     | FU_INFO_RSP                                 | Receiving  |
|                  |                                                                                             | Receive FU_STATE_REQ                                    | FU_STATE_RSP (.State = 1)                   | Receiving  |
|                  |                                                                                             | Receive FU_MISS_REQ                                     | FU_MISS_LIST or FU_MISS_BITMAP              | Receiving  |
|                  |                                                                                             | Receive FU_INIT_REQ                                     | FU_STATE_RSP (.State = 1)                   | Receiving  |
|                  |                                                                                             | Receive FU_DATA                                         | (receiving data, normal behavior)           | Receiving  |
|                  |                                                                                             | Receive FU_EXEC_REQ                                     | FU_STATE_RSP (.State = 1)                   | Receiving  |
|                  |                                                                                             | Receive FU_CONFIRM_REQ                                  | FU_STATE_RSP (.State = 1)                   | Receiving  |
|                  |                                                                                             | Receive FU_KILL_REQ                                     | FU_STATE_RSP (.State = 0); (switch to Idle) | Idle       |
|                  |                                                                                             | Any exception                                           |                                             | Exception  |
| <b>Complete</b>  | Upgrade completed, image integrity ok, the SN is waiting to reboot with the new FW version. | Receive FU_INFO_REQ                                     | FU_INFO_RSP                                 | Complete   |
|                  |                                                                                             | Receive FU_STATE_REQ                                    | FU_STATE_RSP (.State = 2)                   | Complete   |
|                  |                                                                                             | Receive FU_MISS_REQ                                     | FU_STATE_RSP (.State = 2)                   | Complete   |
|                  |                                                                                             | Receive FU_INIT_REQ                                     | FU_STATE_RSP (.State = 2)                   | Complete   |
|                  |                                                                                             | Receive FU_DATA                                         | (ignore)                                    | Complete   |
|                  |                                                                                             | Receive FU_EXEC_REQ with RestartTimer != 0              | FU_STATE_RSP (.State = 3)                   | Countdown  |
|                  |                                                                                             | Receive FU_EXEC_REQ with RestartTimer = 0               | FU_STATE_RSP (.State = 4)                   | Upgrade    |
|                  |                                                                                             | Receive FU_CONFIRM_REQ                                  | FU_STATE_RSP (.State = 2)                   | Complete   |
|                  |                                                                                             | Receive FU_KILL_REQ                                     | FU_STATE_RSP (.State = 0); (switch to Idle) | Idle       |
|                  |                                                                                             | Any exception                                           |                                             | Exception  |
| <b>Countdown</b> |                                                                                             | RestartTimer expires                                    | (switch to Upgrade)                         | Upgrade    |

|                  | Description                                                                                  | Event                                             | Output (or action to be performed)                                              | Next state       |
|------------------|----------------------------------------------------------------------------------------------|---------------------------------------------------|---------------------------------------------------------------------------------|------------------|
|                  | Waiting until <i>RestartTimer</i> expires.                                                   | Receive FU_INFO_REQ                               | FU_INFO_RSP                                                                     | <i>Countdown</i> |
|                  |                                                                                              | Receive FU_STATE_REQ                              | FU_STATE_RSP (.State = 3)                                                       | <i>Countdown</i> |
|                  |                                                                                              | Receive FU_MISS_REQ                               | FU_STATE_RSP (.State = 3)                                                       | <i>Countdown</i> |
|                  |                                                                                              | Receive FU_INIT_REQ                               | FU_STATE_RSP (.State = 3)                                                       | <i>Countdown</i> |
|                  |                                                                                              | Receive FU_DATA                                   | (ignore)                                                                        | <i>Countdown</i> |
|                  |                                                                                              | Receive FU_EXEC_REQ with <i>RestartTimer</i> != 0 | FU_STATE_RSP (.State = 3); (update <i>RestartTimer</i> and <i>SafetyTimer</i> ) | <i>Countdown</i> |
|                  |                                                                                              | Receive FU_EXEC_REQ with <i>RestartTimer</i> = 0  | FU_STATE_RSP (.State = 4); (update <i>RestartTimer</i> and <i>SafetyTimer</i> ) | <i>Upgrade</i>   |
|                  |                                                                                              | Receive FU_CONFIRM_REQ                            | FU_STATE_RSP (.State = 3)                                                       | <i>Countdown</i> |
|                  |                                                                                              | Receive FU_KILL_REQ                               | FU_STATE_RSP (.State = 0); (switch to <i>Idle</i> )                             | <i>Idle</i>      |
|                  |                                                                                              | Any exception                                     |                                                                                 | <i>Exception</i> |
| <b>Upgrade</b>   | The FU mechanism reboots using the new FW image and tests it for <i>SafetyTimer</i> seconds. | <i>SafetyTimer</i> expires                        | FU_STATE_RSP (.State = 4); (switch to <i>Exception</i> , FW rejected)           | <i>Exception</i> |
|                  |                                                                                              | Receive FU_INFO_REQ                               | FU_INFO_RSP                                                                     | <i>Upgrade</i>   |
|                  |                                                                                              | Receive FU_STATE_REQ                              | FU_STATE_RSP (.State = 4)                                                       | <i>Upgrade</i>   |
|                  |                                                                                              | Receive FU_MISS_REQ                               | FU_STATE_RSP (.State = 4)                                                       | <i>Upgrade</i>   |
|                  |                                                                                              | Receive FU_INIT_REQ                               | FU_STATE_RSP (.State = 4)                                                       | <i>Upgrade</i>   |
|                  |                                                                                              | Receive FU_DATA                                   | (ignore)                                                                        | <i>Upgrade</i>   |
|                  |                                                                                              | Receive FU_EXEC_REQ                               | FU_STATE_RSP (.State = 4)                                                       | <i>Upgrade</i>   |
|                  |                                                                                              | Receive FU_CONFIRM_REQ                            | FU_STATE_RSP (.State = 0); (switch to <i>Idle</i> , FW accepted)                | <i>Idle</i>      |
|                  |                                                                                              | Receive FU_KILL_REQ                               | FU_STATE_RSP (.State = 0); (switch to <i>Idle</i> , FW rejected)                | <i>Idle</i>      |
|                  |                                                                                              | Any exception                                     |                                                                                 | <i>Exception</i> |
| <b>Exception</b> | Upon any exception on the firmware upgrade service node will go into this state              | Receive FU_INFO_REQ                               | FU_INFO_RSP                                                                     | <i>Exception</i> |
|                  |                                                                                              | Receive FU_STATE_REQ                              | FU_STATE_RSP (.State = 5)                                                       | <i>Exception</i> |
|                  |                                                                                              | Receive FU_MISS_REQ                               | FU_STATE_RSP (.State = 5)                                                       | <i>Exception</i> |
|                  |                                                                                              | Receive FU_INIT_REQ                               | FU_STATE_RSP (.State = 5)                                                       | <i>Exception</i> |
|                  |                                                                                              | Receive FU_DATA                                   | (ignore)                                                                        | <i>Exception</i> |
|                  |                                                                                              | Receive FU_EXEC_REQ                               | FU_STATE_RSP (.State = 5)                                                       | <i>Exception</i> |

|  | Description | Event                  | Output (or action to be performed) | Next state |
|--|-------------|------------------------|------------------------------------|------------|
|  |             | Receive FU_CONFIRM_REQ | FU_STATE_RSP (.State = 5)          | Exception  |
|  |             | Receive FU_KILL_REQ    | FU_STATE_RSP (.State = 0)          | Idle       |

The state diagram is represented below. Please note that only the most relevant events are shown in the state transitions. See 6.3.5.3 for a detailed description of each state’s behavior and the events and actions related to them. A short description of each state is provided in 6.3.5.2.

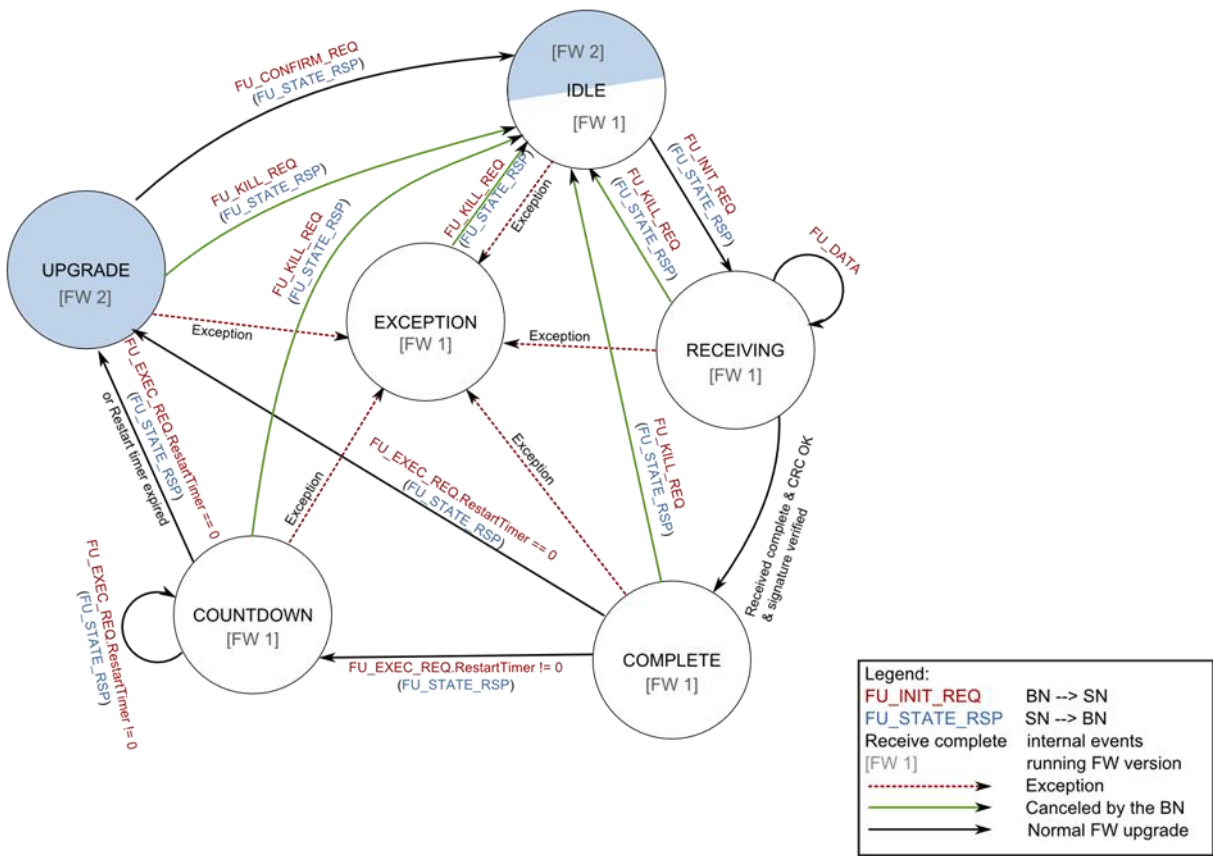


Figure 141 - Firmware Upgrade mechanism, state diagram

6.3.5.2 State description

6.3.5.2.1 Idle

The Service Nodes are in “Idle” state when they are not performing a firmware upgrade. The reception of a FU\_INIT\_REQ message is the only event that forces the Service Node to switch to the next state (“Receiving”). FU\_KILL\_REQ aborts the upgrade process and forces the Service Nodes to switch from any state to “Idle”.

**5373 6.3.5.2.2 Receiving**

5374 The Service Nodes receive the signed firmware via FU\_DATA messages. Service Nodes report complete  
5375 reception of the image answering with either an empty FU\_MISS\_LIST or an empty FU\_MISS\_BITMAP to the  
5376 FU\_MISS\_REQ requests sent by the BN.

5377 If during the reception of the signed firmware the Service Node receives a block with a length that differs  
5378 from the one configured in FU\_INIT\_REQ or a with an packet index out of bounds it should switch to  
5379 "Exception" state with "Protocol" code.

5380 Once the download is complete, a Service Node shall check the integrity of the signed firmware by CRC  
5381 calculation. If the CRC is wrong, the SN shall drop the signed firmware and switch to "Exception" state with  
5382 "CRC verification fail" exception code.

5383 If the CRC results to be ok, the SN shall verify that the signed firmware is correctly signed with the  
5384 manufacturer's key. In case this verification fails, the SN shall drop the signed firmware and switch to  
5385 "Exception" state with "Signature verification fail" exception code.

5386 If the signature is verified successfully, the SN shall switch to "Complete" state.

5387 The CRC check on the complete signed firmware and the later signature verification is mandatory, and is  
5388 automatically started by the SNs. The service node shall not accept any image that is not properly signed.

5389 Note that these checks at SN side are not immediate. There may be a not negligible time interval between  
5390 the message sent by the SN reporting that the reception is complete and the transition to "Complete".

**5391 6.3.5.2.3 Complete**

5392 A Service Node in "Complete" state waits until reception of a FU\_EXEC\_REQ message. The Service Node may  
5393 switch either to "Countdown" or "Upgrade" depending on the field *RestartTimer*, which specifies in which  
5394 instant the Service Node has to reboot using the new firmware. If *RestartTimer* = 0, the Service Node  
5395 immediately switches to "Upgrade"; else, the Service Node switches to "Countdown".

**5396 6.3.5.2.4 Countdown**

5397 A Service Node in "Countdown" state waits a period of time specified in the *RestartTimer* field of a previous  
5398 FU\_EXEC\_REQ message. When this timer expires, it automatically switches to "Upgrade".

5399 FU\_EXEC\_REQ can be used in "Countdown" state to reset *RestartTimer* and *SafetyTimer*. In this case, both  
5400 timers have to be specified in FU\_EXEC\_REQ because both will be overwritten. Note that it is possible to force  
5401 the Node to immediately switch from "Countdown" to "Upgrade" state setting *RestartTimer* to zero.

**5402 6.3.5.2.5 Upgrade**

5403 A Service Node in "Upgrade" state shall run the new firmware during a time period specified in  
5404 FU\_EXEC\_REQ.SafetyTimer.

5405 If it does not receive any confirmation at all before this timer expires, the Service Node discards the new FW,  
5406 reboots with the old version and switches to "Exception" state with "Safety time expired" code.

5407 In case the SN receives a FU\_KILL\_REQ message it will discard the new FW, reboot with the old version and  
5408 switch to “Idle” state.

#### 5409 **6.3.5.2.6 Exception**

5410 A Service Node can enter in exception state from any other state upon an event related to the Firmware  
5411 Upgrade that shall be notified to the Base Node as an exception.

5412 In case the SN receives a FU\_KILL\_REQ in “Exception” state it shall discard any ongoing FW upgrade progress  
5413 and switch to “idle” state. On any other event the SN will take no action and respond a FU\_STATE\_RSP to any  
5414 request with the code describing the specific exception. Exception state has a code, that shall have  
5415 information that can give more information on the exception happened. This code shall set the “temporary”  
5416 flag in case restarting the same Firmware Upgrade process could turn in success.

5417 There is a field up to the manufacturer of one byte for additional information about the exception, the format  
5418 of this field is out of the scope of this specification.

#### 5419 **6.3.5.3 Control packets**

##### 5420 **6.3.5.3.1 FU\_INIT\_REQ**

5421 The Base Node sends this packet in order to configure a Service Node for the Firmware Upgrade. If the Service  
5422 Node is in “Idle” state, it will change its state from “Idle” to “Receiving” and will answer with FU\_STATE\_RSP.  
5423 In any other case it will just answer sending FU\_STATE\_RSP.

5424 The content of FU\_INIT\_REQ is shown below.

5425 **Table 117 - Fields of FU\_INIT\_REQ**

| Field     | Length  | Description                                                                                    |
|-----------|---------|------------------------------------------------------------------------------------------------|
| Type      | 4 bits  | 0 = FU_INIT_REQ.                                                                               |
| Version   | 2 bits  | 0 for this version of the protocol.                                                            |
| PageSize  | 2 bits  | 0 for a PageSize=32;<br>1 for a PageSize=64;<br>2 for a PageSize=128;<br>3 for a PageSize=192. |
| ImageSize | 32 bits | Size of the signed firmware in bytes.                                                          |

| Field               | Length  | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |
|---------------------|---------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| CRC                 | 32 bits | CRC of the signed firmware.<br><br>The input polynomial $M(x)$ is formed as a polynomial whose coefficients are bits of the data being checked (the first bit to check is the highest order coefficient and the last bit to check is the coefficient of order zero). The Generator polynomial for the CRC is $G(x)=x^{32}+x^{26}+x^{23}+x^{22}+x^{16}+x^{12}+x^{11}+x^{10}+x^8+x^7+x^5+x^4+x^2+x+1$ . The remainder $R(x)$ is calculated as the remainder from the division of $M(x) \cdot x^{32}$ by $G(x)$ . The coefficients of the remainder will then be the resulting CRC. |
| Signature algorithm | 4 bits  | 0 – Not specified (Default signature algorithm implemented in Service Node will be used)<br><br>1 – RSA 3072 + SHA-256<br><br>2 – ECDSA 256 + SHA-256<br><br>3-15 – Reserved for future use                                                                                                                                                                                                                                                                                                                                                                                      |
| Reserved            | 2 bits  | Shall be 0 for this version of the document. Reserved for future use.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            |
| Signature length    | 10 bits | Length of the signature part of the signed firmware in bytes, ignored if signature algorithm value is 0.                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |

#### 5426 6.3.5.3.2 FU\_EXEC\_REQ

5427 This packet is used by the Base Node to command a Service Node in “Complete” state to restart using the  
5428 new firmware, once the complete image has been received by the Service Node. FU\_EXEC\_REQ specifies  
5429 when the Service Node has to restart and how long the “safety” period shall be, as explained in 6.3.5.2.5.  
5430 Additionally, FU\_EXEC\_REQ can be used in “Countdown” state to reset the restart and the safety timers.

5431 Depending on the value of *RestartTimer*, a Service Node in “Complete” state may change either to  
5432 “Countdown” or to “Upgrade” state. In any case, the Service Node answers with FU\_STATE\_RSP.

5433 In “Countdown” state, the Base Node can reset *RestartTimer* and *SafetyTimer* with a FU\_EXEC\_REQ message  
5434 (both timers must be specified in the message because both will be overwritten).

5435 The content of this packet is described below.

5436 Table 118 - Fields of FU\_EXEC\_REQ

| Field    | Length | Description                         |
|----------|--------|-------------------------------------|
| Type     | 4 bits | 1 = FU_EXEC_REQ.                    |
| Version  | 2 bits | 0 for this version of the protocol. |
| Reserved | 2 bits | 0 .                                 |



| Field               | Length  | Description                                                                               |
|---------------------|---------|-------------------------------------------------------------------------------------------|
| <i>RestartTimer</i> | 16 bits | 0..65536 seconds; time before restarting with new FW.                                     |
| <i>SafetyTimer</i>  | 16 bits | 0..65536 seconds; time to test the new FW. It starts when the “Upgrade” state is entered. |

5437

5438 **6.3.5.3.3 FU\_CONFIRM\_REQ**

5439 This packet is sent by the Base Node to a Service Node in “Upgrade” state to confirm the current FW. If the  
 5440 Service Node receives this message, it discards the old FW version and switches to “Idle” state. The Service  
 5441 Node answers with FU\_STATE\_RSP when receiving this message.

5442 In any other state, the Service Node answers with FU\_STATE\_RSP without performing any additional actions.

5443 This packet contains the fields described below.

5444 **Table 119 - Fields of FU\_CONFIRM\_REQ**

| Field           | Length | Description                         |
|-----------------|--------|-------------------------------------|
| Type            | 4 bits | 2 = FU_CONFIRM_REQ.                 |
| Version         | 2 bits | 0 for this version of the protocol. |
| <i>Reserved</i> | 2 bits | 0.                                  |

5445 **6.3.5.3.4 FU\_STATE\_REQ**

5446 This packet is sent by the Base Node in order to get the Firmware Upgrade state of a Service Node. The  
 5447 Service Node will answer with FU\_STATE\_RSP.

5448 This packet contains the fields described below.

5449 **Table 120 - Fields of FU\_STATE\_REQ**

| Field           | Length | Description                         |
|-----------------|--------|-------------------------------------|
| Type            | 4 bits | 3 = FU_STATE_REQ.                   |
| Version         | 2 bits | 0 for this version of the protocol. |
| <i>Reserved</i> | 2 bits | 0.                                  |

5450

5451 **6.3.5.3.5 FU\_KILL\_REQ**

5452 The Base Node sends this message to terminate the Firmware Upgrade process. A Service Node receiving this  
 5453 message will automatically switch to “Idle” state and optionally delete the downloaded data. The Service  
 5454 Node replies sending FU\_STATE\_RSP.

5455 The content of this packet is described below.

5456 **Table 121 - Fields of FU\_KILL\_REQ**

| Field    | Length | Description                         |
|----------|--------|-------------------------------------|
| Type     | 4 bits | 4 = FU_KILL_REQ.                    |
| Version  | 2 bits | 0 for this version of the protocol. |
| Reserved | 2 bits | 0.                                  |

### 5457 6.3.5.3.6 FU\_STATE\_RSP

#### 5458 6.3.5.3.6.1 General

5459 This packet is sent by the Service Node as an answer to FU\_STATE\_REQ, FU\_KILL\_REQ, FU\_EXEC\_REQ,  
 5460 FU\_CONFIRM\_REQ or FU\_INIT\_REQ messages received through the unicast connection. It is used to notify  
 5461 the Firmware Upgrade state in a Service Node.

5462 Additionally, FU\_STATE\_RSP is used as default response to all events that happen in states where they are  
 5463 not foreseen (e.g. FU\_EXEC\_REQ in “Receiving” state, FU\_INIT\_REQ in “Upgrade” ...).

5464 This packet contains the fields described below.

5465 **Table 122 - Fields of FU\_STATE\_RSP**

| Field    | Length  | Description                                                                                                                                     |
|----------|---------|-------------------------------------------------------------------------------------------------------------------------------------------------|
| Type     | 4 bits  | 5 = FU_STATE_RSP.                                                                                                                               |
| Version  | 2 bits  | 0 for this version of the protocol.                                                                                                             |
| Reserved | 2 bits  | 0.                                                                                                                                              |
| State    | 4 bits  | 0 for Idle;<br>1 for Receiving;<br>2 for Complete;<br>3 for Countdown;<br>4 for Upgrade;<br>5 for Exception<br>6 to 15 reserved for future use. |
| Reserved | 4 bits  | 0.                                                                                                                                              |
| CRC      | 32 bits | CRC as the one received in the CRC field of FU_INIT_REQ.                                                                                        |

| Field          | Length  | Description                                                                                                                                                                 |
|----------------|---------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Received       | 32 bits | Number of received pages (this field should only be present if State is Receiving).                                                                                         |
| Exception code | 16 bits | Exception code describing the exception occurred (this field shall only be present if State is Exception)<br><br>This field is described with detail in section 6.3.5.3.6.2 |

#### 6.3.5.3.6.2 Exception code

The exception code has a number of fields that give more information to the Base Node about the exception that have happened during the Firmware Upgrade process.

Table 123 – Fields of Exception code

| Field             | Length | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |
|-------------------|--------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Permanent         | 1 bit  | Flag used to inform if a retry on the firmware upgrade will not success, because the exception being permanent.<br><br>0 if the exception is temporary<br><br>1 if the exception is permanent                                                                                                                                                                                                                                                                                                                                                                                   |
| Code              | 7 bits | Code describing the type of exception that happened<br><br><b>0 – General:</b> for an exception that do not fit any of the other codes<br><br><b>1 – Protocol:</b> Page number out of bounds, page length mismatch from FU_INIT_REQ...<br><br><b>2 – CRC verification fail:</b> If the CRC verification failed<br><br><b>3 – Invalid image:</b> If the image was not a firmware image or not for the device<br><br><b>4 – Signature verification fail:</b> the signature verification failed.<br><br><b>5 – Safety time expired:</b> the safety time in "upgrade" state expires |
| Manufacturer code | 8 bits | Field that provides additional detail about the exception.<br><br>This code is up to the manufacturer.                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |

#### 6.3.5.3.7 FU\_DATA

This packet is sent by the Base Node to transfer a page of the signed firmware to a Service Node. No answer is expected by the Base Node.

This packet contains the fields described below.

Table 124 - Fields of FU\_DATA

| Field           | Length          | Description                                                                                                                                                                  |
|-----------------|-----------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Type            | 4 bits          | 6 = FU_DATA.                                                                                                                                                                 |
| Version         | 2 bits          | 0 for this version of the protocol.                                                                                                                                          |
| <i>Reserved</i> | 2 bits          | 0.                                                                                                                                                                           |
| PageIndex       | 32 bits         | Index of the page being transmitted.                                                                                                                                         |
| <i>Reserved</i> | 8 bits          | Padding byte for 16-bit devices. Set to 0 by default.                                                                                                                        |
| Data            | <i>Variable</i> | Data of the page.<br><br>The length of this data is PageSize (32, 64, 128 or 192) bytes for every page, except the last one that will have the remaining bytes of the image. |

#### 5475 6.3.5.3.8 FU\_MISS\_REQ

5476 This packet is sent by the Base Node to a Service Node to request information about the pages that are still  
5477 to be received.

5478 If the Service Node is in “Receiving” state it will answer with a FU\_MISS\_BITMAP or FU\_MISS\_LIST message.

5479 If the Service Node is in any other state it will answer with a FU\_STATE\_RSP.

5480 This packet contains the fields described below.

5481 Table 125 - Fields of FU\_MISS\_REQ

| Field           | Length  | Description                                               |
|-----------------|---------|-----------------------------------------------------------|
| Type            | 4 bits  | 7 = FU_MISS_REQ.                                          |
| Version         | 2 bits  | 0 for this version of the protocol.                       |
| <i>Reserved</i> | 2 bits  | 0.                                                        |
| PageIndex       | 32 bits | Starting point to gather information about missing pages. |

#### 5482 6.3.5.3.9 FU\_MISS\_BITMAP

5483 This packet is sent by the Service Node as an answer to a FU\_MISS\_REQ. It carries the information about the  
5484 pages that are still to be received.

5485 This packet will contain the fields described below.

5486 Table 126 - Fields of FU\_MISS\_BITMAP

| Field | Length | Description         |
|-------|--------|---------------------|
| Type  | 4 bits | 8 = FU_MISS_BITMAP. |

| Field           | Length          | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |
|-----------------|-----------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Version         | 2 bits          | 0 for this version of the protocol.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |
| <i>Reserved</i> | 2 bits          | 0.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               |
| <i>Received</i> | 32bits          | Number of received pages.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |
| PageIndex       | 32 bits         | Page index of the page represented by the first bit of the bitmap. It should be the same as the <i>PageIndex</i> field in FU_MISS_REQ messages, or a posterior one. If it is posterior, it means that the pages in between are already received. In this case, if all pages after the <i>PageIndex</i> specified in FU_MISS_REQ have been received, the Service Node shall start looking from the beginning ( <i>PageIndex</i> = 0).                                                                                                                                                                                             |
| Bitmap          | <i>Variable</i> | <p>This bitmap contains the information about the status of each page.</p> <p>The first bit (most significant bit of the first byte) represents the status of the page specified by <i>PageIndex</i>. The next bit represents the status of the <i>PageIndex+1</i> and so on.</p> <p>A '1' represents that a page is missing, a '0' represents that the page is already received.</p> <p>After the bit that represents the last page in the image, it is allowed to overflow including bits that represent the missing status of the page with index zero.</p> <p>The maximum length of this field is <i>PageSize</i> bytes.</p> |

5487

5488 It is up to the Service Node to decide to send this type of packet or a FU\_MISS\_LIST message. It is usually  
5489 more efficient to transmit this kind of packets when the number of missing packets is not very low. But it is  
5490 up to the implementation to transmit one type of packet or the other. The Base Node should understand  
5491 both.

5492 In case a Service Node receives a FU\_MISS\_REQ during CRC calculation, it shall respond either with an empty  
5493 FU\_MISS\_BITMAP or an empty FU\_MISS\_LIST.

#### 5494 6.3.5.3.10 FU\_MISS\_LIST

5495 This packet is sent by the Service Node as an answer to a FU\_MISS\_REQ. It carries the information about the  
5496 pages that are still to be received.

5497 This packet will contain the fields described below.

5498

Table 127 - Fields of FU\_MISS\_LIST

| Field | Length | Description       |
|-------|--------|-------------------|
| Type  | 4 bits | 9 = FU_MISS_LIST. |

| Field           | Length   | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |
|-----------------|----------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Version         | 2 bits   | 0 for this version of the protocol.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |
| <i>Reserved</i> | 2 bits   | 0.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               |
| <i>Received</i> | 32 bits  | Number of received pages.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |
| PageIndexList   | Variable | <p>List of pages that are still to be received. Each page is represented by its PageIndex, coded as a 32 bit integer.</p> <p>These pages should be sorted in ascending order (low to high), being possible to overflow to the <i>PageIndex</i> equal to zero to continue from the beginning.</p> <p>The first page index should be the same as the <i>PageIndex</i> field in FU_MISS_REQ, or a posterior one. If it is posterior, it means that the pages in between are already received (by posterior it is allowed to overflow to the page index zero, to continue from the beginning).</p> <p>The maximum length of this field is <i>PageSize</i> bytes.</p> |

5499 It is up to the Service Node to decide to transmit this packet type or a FU\_MISS\_BITMAP message. It is usually  
5500 more efficient to transmit this kind of packets when the missing packets are very sparse, but it is  
5501 implementation-dependent to transmit one type of packet or the other. The Base Node should understand  
5502 both.

5503 In case a Service Node receives a FU\_MISS\_REQ during CRC calculation, it shall respond either with an empty  
5504 FU\_MISS\_BITMAP or an empty FU\_MISS\_LIST.

#### 5505 6.3.5.3.11 FU\_INFO\_REQ

5506 This packet is sent by a Base Node to request information from a Service Node, such as manufacturer, device  
5507 model, firmware version and other parameters specified by the manufacturer. The Service Node will answer  
5508 with one or more FU\_INFO\_RSP packets.

5509 This packet contains the fields described below.

5510 **Table 128 - Fields of FU\_INFO\_REQ**

| Field           | Length   | Description                                                                                         |
|-----------------|----------|-----------------------------------------------------------------------------------------------------|
| Type            | 4 bits   | 10 = FU_INFO_REQ.                                                                                   |
| Version         | 2 bits   | 0 for this version of the protocol.                                                                 |
| <i>Reserved</i> | 2 bits   | 0.                                                                                                  |
| InfoldList      | Variable | <p>List of identifiers with the information to retrieve.</p> <p>Each identifier is 1 byte long.</p> |

| Field | Length | Description                                   |
|-------|--------|-----------------------------------------------|
|       |        | The maximum length of this field is 32 bytes. |

5511

5512 The following identifiers are defined:

5513

Table 129 - Infold possible values

| Infold  | Name                         | Description                                     |
|---------|------------------------------|-------------------------------------------------|
| 0       | Manufacturer                 | Universal Identifier of the Manufacturer.       |
| 1       | Model                        | Model of the product working as Service Node.   |
| 2       | Firmware                     | Current firmware version being executed.        |
| 128-255 | <i>Manufacturer specific</i> | Range of values that are manufacturer specific. |

5514

5515 **6.3.5.3.12 FU\_INFO\_RSP**

5516 This packet is sent by a Service Node as a response to a FU\_INFO\_REQ message from the Base Node. A Service  
 5517 Node may have to send more than one FU\_INFO\_RSP when replying to a information request by the Base  
 5518 Node.

5519 This packet contains the fields described below.

5520

Table 130 - Fields of FU\_INFO\_RSP

| Field           | Length        | Description                                                                                                                                                                                                                   |
|-----------------|---------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Type            | 4 bits        | 11 = FU_INFO_RSP.                                                                                                                                                                                                             |
| Version         | 2 bits        | 0 for this version of the protocol.                                                                                                                                                                                           |
| <i>Reserved</i> | 2 bits        | 0.                                                                                                                                                                                                                            |
| InfoData        | 0 – 192 bytes | Data with the information requested by the Base Node. It may contain several entries (one for each requested identifier), each entry has a maximum size of 32 bytes. The maximum size of this field is 192 bytes (6 entries). |

5521

5522 The InfoData field can contain several entries, the format of each entry is specified below.

5523

5524 Table 131 - Fields of each entry of InfoData in FU\_INFO\_RSP

| Field           | Length       | Description                                                                                                                                                     |
|-----------------|--------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Infold          | 8 bits       | Identifier of the information as specified in 6.3.5.3.11.                                                                                                       |
| <i>Reserved</i> | 3 bits       | 0.                                                                                                                                                              |
| Length          | 5 bits       | Length of the Data field (If Length is 0 it means that the specified Infold is not supported by the specified device).                                          |
| Data            | 0 – 30 bytes | Data with the information provided by the Service Node.<br><br>Its content may depend on the meaning of the Infold field. No value may be longer than 30 bytes. |

#### 5525 6.3.5.4 Firmware integrity and authentication

##### 5526 6.3.5.4.1 General

5527 The firmware integrity and authentication is ensured by two means: the firmware CRC and the firmware  
5528 signature. Both CRC and signature verifications are performed in the state “Receiving”, on the complete  
5529 image after the receiving completion.

##### 5530 6.3.5.4.2 Image CRC

5531 The role of the firmware upgrade CRC is to check the integrity of the image received from the Base Node,  
5532 over the link Base Node – Service Node.

5533 The Base node, before initiating the firmware upgrade process, calculates a 32 bits CRC on the complete  
5534 image, using the Generator polynomial  $G(x)=x^{32}+x^{26}+x^{23}+x^{22}+x^{16}+x^{12}+x^{11}+x^{10}+x^8+x^7+x^5+x^4+x^2+x+1$ ,  
5535 and appends the result at the end of the firmware upgrade payload.

5536 After the receiving completion, the Service Node must verify the integrity of the whole image by the CRC  
5537 recalculation. The firmware upgrade is deemed valid only when this CRC recalculation is successful.

##### 5538 6.3.5.4.3 Image signature

5539 The role of the signature is to verify the integrity and the authenticity of the image as generated by the  
5540 manufacturer. Indeed the firmware image, as generated by the image originator (the chip manufacturer) is  
5541 provided to the Base Node via several different means. The firmware image signature provides evidence that  
5542 the firmware image was not altered or substituted along all these transportation means and locations, and  
5543 evidence that the concerned manufacturer is the originator.

5544 The signature is generated by the originator of the firmware image, on the whole image using asymmetric  
5545 key cryptography, and then included in the signed firmware to be delivered. The algorithm used for this  
5546 purpose, how to equip the Service Nodes with the public key, and the infrastructure for certificate  
5547 management are the scope of the firmware image generator. The algorithm used must comply with FIPS 186-  
5548 4 standard, <http://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.186-4.pdf>, preferably ECDSA 256 bits or RSA  
5549 3072 bits as an alternate solution.



5550 Firmware image originators are strongly recommended for equipping the entities having in charge the  
 5551 spreading of the firmware image with tools allowing them to process to the verification before the  
 5552 deployment.

5553 After receiving the signed firmware the Service node must verify the firmware image signature. The image is  
 5554 deemed valid only when the verification is successful.

### 5555 6.3.6 Examples

5556 The figures below are an example of the traffic generated between the Base Node and the Service Node  
 5557 during the Firmware Upgrade process.

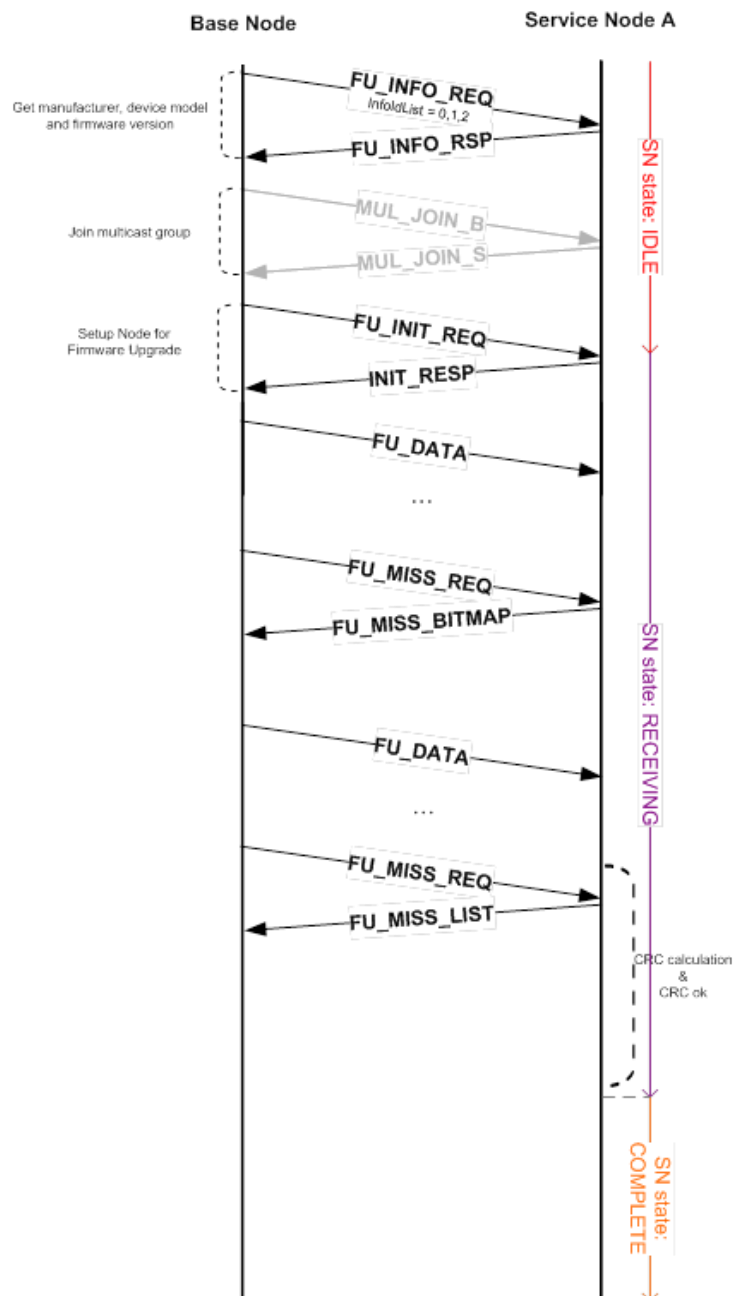


Figure 142 - Init Service Node and complete FW image

Figure 142 shows the initialization of the process, the FW download and the integrity check of the image. In the example above, the downloaded FW image is supposed to be complete before sending the last FU\_MISS\_REQ. The Base Node sends it to verify its bitmap. In this example, FU\_MISS\_LIST has an empty PageIndexList field, which means that the FW image is complete.

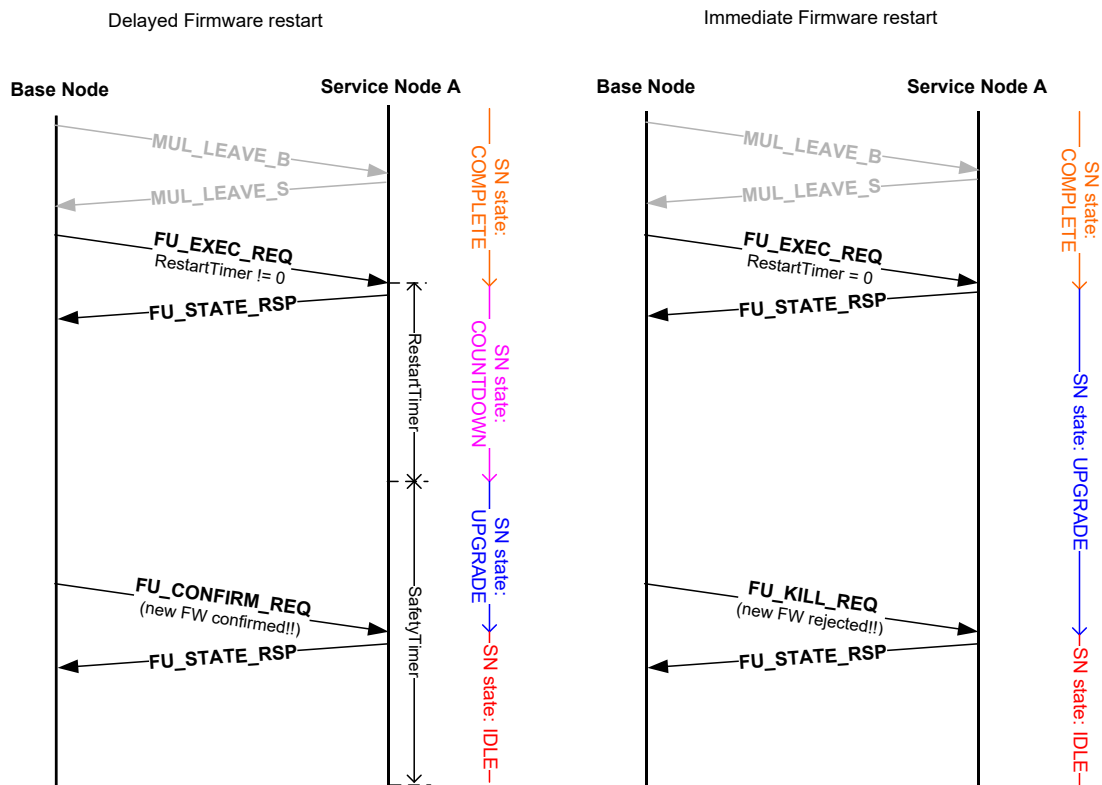


Figure 143 - Execute upgrade and confirm/reject new FW version

Above it is shown how to proceed after completing the FW download. The Base Node commands the Service Node to reboot either immediately ("Immediate Firmware Start", RestartTimer = 0) or after a defined period of time ("Delayed Firmware start", RestartTimer != 0). After reboot, the Base Node can either confirm the recently downloaded message sending a FU\_CONFIRM\_REQ or reject it (sending a FU\_KILL\_REQ or letting the safety period expire doing nothing).

## 6.4 Management interface description

### 6.4.1 General

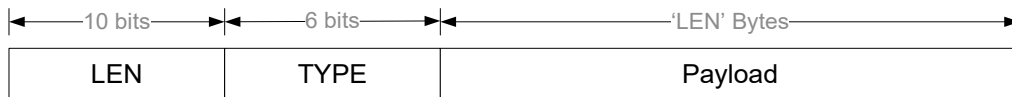
Management functions defined in earlier sections shall be available over an abstract management interface specified in this section. The management interface can be accessed over diverse media. Each physical media shall specify its own management plane communication profile over which management information is exchanged. It is mandatory for implementations to support PRIME management plane communication profile. All other "management plane communication profiles" are optional and maybe mandated by certain "application profiles" to use in specific cases.

The present version of specifications describes two communication profiles, one of which is over this specification NULL SSCS and other over serial link.

5581 With these two communication profiles, it shall be possible to address the following use-cases:

- 5582       • Remote access of management interface over NULL SSCS. This shall enable Base Node's use as a  
5583       supervisory gateway for all devices in a Subnetwork
- 5584       • Local access of management interface (over peripherals like RS232, USBSerial etc) in a Service  
5585       Node. Local access shall fulfill cases where a coprocessor exists for supervisory control of  
5586       processor or when manual access is required over local physical interface for maintenance.

5587 Management data comprises of a 2 bytes header followed by payload information corresponding to the type  
5588 of information carried in message. The header comprises of a 10 bit length field and 6 bit message\_id field.



5589

5590 **Figure 144 – Management data frame**

5591 **Table 132 - Management data frame fields**

| Name      | Length  | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               |
|-----------|---------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| MGMT.LEN  | 10 bits | <p>Length of payload data following the 2 byte header.</p> <p>LEN=0 implies there is no payload data following this header and the TYPE field contains all required information to perform appropriate action.</p> <p>NOTE: The length field maybe redundant in some communication profiles (e.g. When transmitted over PRIME), but is required in others. Therefore for the sake of uniformity, it is always included in management data.</p>                                                                                                                                                                                                                                                                                                                                                            |
| MGMT.TYPE | 6 bits  | <p>Type of management information carried in corresponding data. Some message_id have standard semantics which should be respected by all PRIME compliant devices while others are reserved for local use by vendors.</p> <p>0x00 – Get PIB attribute query;<br/>           0x01 – Get PIB attribute response;<br/>           0x02 – Set PIB attribute command;<br/>           0x03 – Reset all PIB statistics attributes;<br/>           0x04 – Reboot destination device;<br/>           0x05 – Firmware upgrade protocol message;<br/>           0x06 – Enhanced PIB Query<br/>           0x07 – Enhances PIB Response<br/>           0x08 to 0x0F: Reserved for future use. Vendors should not use these values for local purpose;<br/>           0x10 – 0x3F : Reserved for vendor specific use.</p> |

## 6.4.2 Payload format of management information

### 6.4.2.1 Get PIB attribute query

This query is issued by a remote management entity that is interested in knowing values of PIB attributes maintained on a compliant device with this specification.

The payload may comprise of a query on either a single PIB attribute or multiple attributes. For reasons of efficiency queries on multiple PIB attributes maybe aggregated in one single command. Given that the length of a PIB attribute identifier is constant, the number of attributes requested in a single command is derived from the overall MGMT.LEN field in header.

The format of payload information is shown in the following figure.



Figure 145 – Get PIB Attribute query. Payload

Fields of a GET request are summarized in table below:

Table 133 - GET PIB Attribute request fields

| Name             | Length  | Description                                                                                                                                                                                                                                      |
|------------------|---------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| PIB Attribute id | 2 bytes | 16 bit PIB attribute identifier                                                                                                                                                                                                                  |
| Index            | 1 byte  | Index of entry to be returned for corresponding PIB Attribute id. This field is only of relevance while returning PIB list attributes.<br><br>Index = 0; if PIB Attribute is not a list;<br>Index = 1 to 255; Return list record at given index. |

### 6.4.2.2 Get PIB attribute response

This data is sent out from a compliant device of this specification in response to a query of one or more PIB attributes. If a certain queried PIB attribute is not maintained on the device, it shall still respond to the query with value field containing all '1s' in the response.

The format of payload is shown in the following figure.

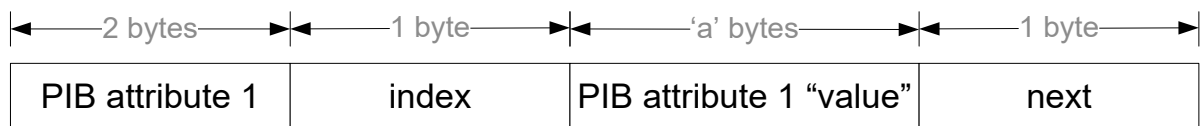


Figure 146 - Get PIB Attribute response. Payload

Fields of a GET request are summarized in table below:

5615 **Table 134 - GET PIB Attribute response fields**

| Name                | Length    | Description                                                                                                                                                                                                                                                                                                                                                                                     |
|---------------------|-----------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| PIB Attribute id    | 2 bytes   | 16 bit PIB attribute identifier.                                                                                                                                                                                                                                                                                                                                                                |
| Index               | 1 byte    | Index of entry returned for corresponding PIB Attribute id. This field is only of relevance while returning PIB list attributes.<br><br>index = 0; if PIB Attribute is not a list.<br><br>index = 1 to 255; Returned list record is at given index.                                                                                                                                             |
| PIB Attribute value | 'a' bytes | Values of requested PIB attribute. In case of a list attribute, value shall comprise of entire record corresponding to given index of PIB attribute                                                                                                                                                                                                                                             |
| Next                | 1 byte    | Index of next entry returned for corresponding PIB Attribute id. This field is only of relevance while returning PIB list attributes.<br><br>next = 0; if PIB Attribute is not a list or if no records follow the one being returned for a list PIB attribute i.e. given record is last entry in list.<br><br>next = 1 to 255; index of next record in list maintained for given PIB attribute. |

5616

5617 Response to PIB attribute query can span across several MAC GPDU's. This shall always be the case when an

5618 aggregated (comprising of several PIB attributes) PIB query's response if longer than the maximum segment

5619 size allowed to be carried over the NULL SCSS.

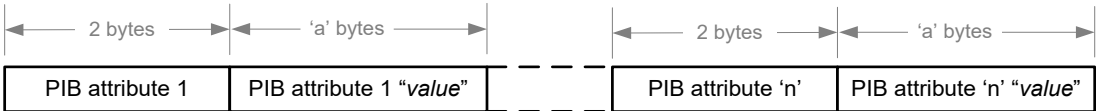
5620 **6.4.2.3 Set PIB attribute**

5621 This management data shall be used to set specific PIB attributes. Such management payload comprises of a

5622 2 byte PIB attribute identifier, followed by the relevant length of PIB attribute information corresponding to

5623 that identifier. For reasons of efficiency, it shall be possible to aggregate SET command on several PIB

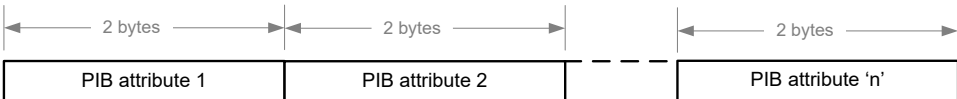
5624 attributes in one GPDU. The format of such an aggregated payload is shown in figure below:



5626 **Figure 147 – Set PIB attribute. Aggregated payload**

5627 For cases where the corresponding PIB attribute is only a trigger (all ACTION PIB attributes), there shall be

5628 no associated value and the request data format shall be as shown below.



5630 **Figure 148 – Set PIB Attribute. ACTION PIB attributes**

5631 It is assumed that the management entity sending out this information has already determined if the  
5632 corresponding attributes are supported at target device. This may be achieved by a previous query and its  
5633 response.

#### 5634 6.4.2.4 Reset statistics

5635 This command has optional payload. In case there is no associated payload, the receiving device shall reset  
5636 all of its PIB statistical attributes.

5637 For cases when a remote management entity only intends to perform reset of selective PIB statistical  
5638 attributes, the payload shall contain a list of attributes that need to be reset. The format shall be the same  
5639 as shown in Section 6.4.2.1.

5640 Since there is no confirmation message going back from the device complying with this specification, the  
5641 management entity needs to send a follow-up PIB attribute query, in case it wants to confirm successful  
5642 completion of appropriate action.

#### 5643 6.4.2.5 Reboot device

5644 There is no corresponding payload associated with this command. The command is complete in itself. The  
5645 receiving compliant device with this specification shall reboot itself on receipt of this message.

5646 It is mandatory for all implementations compliant with this specification to support this command and its  
5647 corresponding action.

#### 5648 6.4.2.6 Firmware upgrade

5649 The payload in this case shall comprise of firmware upgrade commands and responses described in section  
5650 6.2.3.2 of the specification.

#### 5651 6.4.2.7 Enhanced PIB query

##### 5652 6.4.2.7.1 General

5653 This command let perform a variety of queries grouped in one. At the moment there is one query type that  
5654 can be performed, more queries will be added in future releases of the specification.

5655 The format of this command is shown in the Figure 149.

|      |         |         |     |         |
|------|---------|---------|-----|---------|
| 0x06 | Query 1 | Query 2 | ... | Query n |
|------|---------|---------|-----|---------|

5656 Figure 149 – Enhanced PIB query format

|       |                     |        |
|-------|---------------------|--------|
| 1 bit | 7 bits              | 8 bits |
| 0     | Iterator of 15 bits |        |

5657 Figure 150 – Iterator short format

|       |                       |                     |
|-------|-----------------------|---------------------|
| 1 bit | 7 bits                | n bytes             |
| 1     | Iterator length ( n ) | Iterator of n bytes |

5658 Figure 151 – Iterator long format

5659 The available queries are the ones listed in the Table 132.

#### 6.4.2.7.2 PIB list query

This query is used to request the next elements in a collection of elements on a PIB element. Such as node list or connection list.

The format of this query is listed in Table 135

Table 135 – PIB List query format

| Element      | Size(bytes) | Description                                                                                                                                                                                                       |
|--------------|-------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 0x0E         | 1           | Code for the PIB list query operation                                                                                                                                                                             |
| Attribute ID | 2           | Attribute ID of the PIB                                                                                                                                                                                           |
| Number       | 1           | Maximum number of records to retrieve                                                                                                                                                                             |
| Iterator     | *           | Iterator returned in the last item if the PIB list response message, the response will start in the next element after that one.<br><br>The constant value "0x0000" in this field will retrieve the first element |

#### 6.4.2.8 Enhanced PIB response

##### 6.4.2.8.1 General

This command let respond to a variety of queries requested in Enhanced PIB query. At the moment there is one response type, more response types will be added in future releases of the specification.

The format of this command is shown in the Figure 152.

|      |            |            |     |            |
|------|------------|------------|-----|------------|
| 0x07 | Response 1 | Response 2 | ... | Response n |
|------|------------|------------|-----|------------|

Figure 152 – Enhanced PIB response format

The available responses are listed in the Table 132.

##### 6.4.2.8.2 PIB list response

This response is used to send information on lists of PIB collection elements, such as node list or connection list.

The format of this command is shown in the Table 136.

Table 136 – PIB list response format

| Element      | Size(bytes) | Description                                 |
|--------------|-------------|---------------------------------------------|
| 0x0F         | 1           | Code for the PIB list response operation    |
| Attribute ID | 2           | Attribute ID of the PIB                     |
| Number       | 1           | Number of records contained in this message |

| Element     | Size(bytes) | Description                                                       |
|-------------|-------------|-------------------------------------------------------------------|
| End of List | 1           | It defines if the end of the list has been reached (1) or not (0) |
| Length      | 1           | Length of each record                                             |
| Iterator 1  | *           | Iterator of the record #1                                         |
| Value 1     | *           | Value of the record #1                                            |
| ....        | ....        | ....                                                              |
| Iterator n  | *           | Iterator of the record #n                                         |
| Value n     | *           | Value of the record #n                                            |

5677  
5678 The iterator has the same format as the ones described in section 6.4.2.7.2

### 5679 6.4.3 NULL SSCS communication profile

5680 This communication profile enables exchange of management information described in previous sections  
5681 over the NULL SSCS.

5682 The management entities at both transmitting and receiving ends are applications making use of the NULL  
5683 SSCS enumerated in Section 0 of this specs. Data is therefore exchanged as MAC Generic PDUs.

### 5684 6.4.4 Serial communication profile

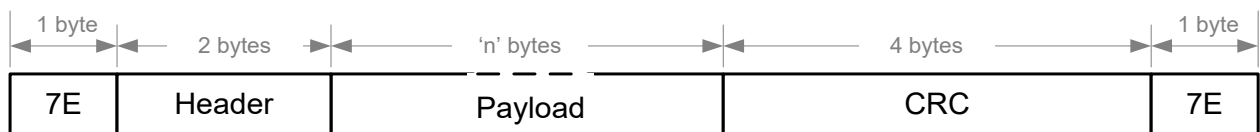
#### 5685 6.4.4.1 Physical layer

5686 The PHY layer maybe any serial link (e.g. RS232, USB Serial). The serial link is required to work 8N1  
5687 configuration at one of the following data rates:

5688 9600 bps, 19200 bps, 38400 bps, 57600 bps.

#### 5689 6.4.4.2 Data encapsulation for management messages

5690 In order ensure robustness, the stream of data is encapsulated in HDLC-type frames which include a 2 byte  
5691 header and 4 byte CRC. All data is encapsulated between a starting flag-byte 0x7E and ending flag-byte 0x7E  
5692 as shown in Figure below:



5693  
5694 **Figure 153 – Data encapsulations for management messages**

5695 If any of the intermediate data characters has the value 0x7E, it is preceded by an escape byte 0x7D, followed  
5696 by a byte derived from XORing the original character with byte 0x20. The same is done if there is a 0x7D  
5697 within the character stream. An example of such case is shown here:



5698

5699 Msg to Tx: 0x01 0x02 0x7E 0x03 0x04 0x7D 0x05 0x06

5700 Actual Tx sequence: 0x01 0x02 0x7D 0x5E 0x03 0x04 0x7D 0x5D 0x05 0x06

5701 Escape Escape

5702 sequence sequence

5703 The 32 bit CRC at end of the frame covers both 'Header' and 'Payload' fields. The CRC is calculated over the  
5704 original data to be transmitted i.e. before byte stuffing of escape sequences described above is performed.  
5705 CRC calculation is

5706 The input polynomial  $M(x)$  is formed as a polynomial whose coefficients are bits of the data being checked  
5707 (the first bit to check is the highest order coefficient and the last bit to check is the coefficient of order zero).  
5708 The Generator polynomial for the CRC is  $G(x)=x^{32}+x^{26}+x^{23}+x^{22}+x^{16}+x^{12}+x^{11}+x^{10}+x^8+x^7+x^5+x^4+x^2+x+1$ .  
5709 The remainder  $R(x)$  is calculated as the remainder from the division of  $M(x) \cdot x^{32}$  by  $G(x)$ . The coefficients of  
5710 the remainder will then be the resulting CRC.

## 5711 6.4.5 TCP communication profile

5712 This communication profile enables exchange of management information described in previous sections  
5713 over a TCP socket. The socket number will be defined by the manufacturer and the device will have the role  
5714 of a TCP server.

5715 Management messages as described in Figure 144 are used in sequence with no extra header in this profile.  
5716 The message boundaries will be described with the length field of the management message.

## 5717 6.5 List of mandatory PIB attributes

### 5718 6.5.1 General

5719 PIB attributes listed in this section shall be supported by all implementations. PIB attributes that are not listed  
5720 in this section are optional and vendors may implement them at their choice. In addition to the PIB attributes,  
5721 the management command to reboot a certain device (as specified in 6.4.2.5) shall also be universally  
5722 supported.

### 5723 6.5.2 Mandatory PIB attributes common to all device types

#### 5724 6.5.2.1 PHY PIB attribute

5725 (See Table 102)

5726 Table 137 - PHY PIB common mandatory attributes

| Attribute Name       | Id     |
|----------------------|--------|
| phyStatsRxTotalCount | 0x00A4 |

| Attribute Name         | Id     |
|------------------------|--------|
| phyStatsBlkAvgEvm      | 0x00A5 |
| phyEmaSmoothing        | 0x00A8 |
| phyRFStatsRxTotalCount | 0x101D |

### 6.5.2.2 MAC PIB attributes

(See Table 106, Table 108 and Table 109)

Table 138 - MAC PIB comon mandatory attributes

| Attribute Name                           | Id     |
|------------------------------------------|--------|
| macEMASmoothing                          | 0x0019 |
| macCSMAR1 (non-Robust Modes only)        | 0x0034 |
| macCSMAR2 (non-Robust Modes only)        | 0x0035 |
| macCSMAR1Robust (Robust Modes supported) | 0x003B |
| macCSMAR2Robust (Robust Modes supported) | 0x003C |

| Attribute Name  | Id     |
|-----------------|--------|
| MacCapabilities | 0x002C |

| List Attribute Name | Id     |
|---------------------|--------|
| macListPhyComm      | 0x0059 |

### 6.5.2.3 Application PIB attributes

(See Table 114)

Table 139 - Applications PIB common mandatory attributes

| Attribute Name | Id     |
|----------------|--------|
| AppFwVersion   | 0x0075 |
| AppVendorId    | 0x0076 |
| AppProductId   | 0x0077 |

## 6.5.3 Mandatory Base Node attributes

### 6.5.3.1 MAC PIB attributes

(See Table 106 and Table 110)

Table 140 - MAC PIB Base Node mandatory attributes

| Attribute Name     | Id     |
|--------------------|--------|
| macBeaconsPerFrame | 0x0013 |

| List Attribute Name | Id     |
|---------------------|--------|
| macListRegDevices   | 0x0050 |
| macListActiveConn   | 0x0051 |

## 6.5.4 Mandatory Service Node attributes

### 6.5.4.1 MAC PIB attributes

(See Table 108, Table 110 and Table 113)

Table 141 - MAC PIB Service Node mandatory attributes

| Attribute Name        | Id     |
|-----------------------|--------|
| macLNID               | 0x0020 |
| MacLSID               | 0x0021 |
| MacSID                | 0x0022 |
| MacSNA                | 0x0023 |
| MacState              | 0x0024 |
| MacSCPLength          | 0x0025 |
| MacNodeHierarchyLevel | 0x0026 |
| MacBeaconSlotCount    | 0x0027 |
| macBeaconRxSlot       | 0x0028 |
| MacBeaconTxSlot       | 0x0029 |
| MacBeaconRxFrequency  | 0x002A |
| MacBeaconTxFrequency  | 0x002B |

5744

| List Attribute Name      | Id     |
|--------------------------|--------|
| macListSwitchTable       | 0x0053 |
| macListAvailableSwitches | 0x0056 |

5745

| Attribute Name      | Id     |
|---------------------|--------|
| MACActionTxData     | 0x0060 |
| MACActionConnClose  | 0x0061 |
| MACActionRegReject  | 0x0062 |
| MACActionProReject  | 0x0063 |
| MACActionUnregister | 0x0064 |
| macSecDUK           | 0x005B |

5746 **6.5.4.2 Application PIB attributes**  
5747 (See Table 115)

5748 Table 142 - APP PIB Service Node mandatory attributes

| Attribute Name           | Id     |
|--------------------------|--------|
| <u>AppFwdlRunning</u>    | 0x0070 |
| <u>AppFwdlRxPktCount</u> | 0x0071 |

5749

Annex A  
(informative)  
Examples of CRC

CRC-8 Example

The table below gives the CRC-8 examples (see section 3.3.2.3) calculated for several specified strings

Table 143 - Examples of CRC-8 calculated for various ASCII strings

| String      | CRC-8 |
|-------------|-------|
| 'T'         | 0xab  |
| "THE"       | 0xa0  |
| 0x03, 0x73  | 0x61  |
| 0x01, 0x3f  | 0xa8  |
| "123456789" | 0xf4  |

CRC-32 Example

The table below gives the CRC-32 example (see section 3.3.2.3)

Table 144 - Example of CRC-32

| String                                                                                                                                                                                                                                                                | CRC-32     |
|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|------------|
| 0x30 0x31 0x32 0x33 0x34 0x35 0x36 0x37 0x38 0x39<br>0x30 0x31 0x32 0x33 0x34 0x35 0x36 0x37 0x38 0x39<br>0x30 0x31 0x32 0x33 0x34 0x35 0x36 0x37 0x38 0x39<br>0x30 0x31 0x32 0x33 0x34 0x35 0x36 0x37 0x38 0x39<br>0x30 0x31 0x32 0x33 0x34 0x35 0x36 0x37 0x38 0x39 | 0x24a56cf5 |

CRC-32 Concatenation Example

The following table shows an example concatenation of SNA with a GPDU and resulting CRC-32 (see section 4.4.2.4).

Considering:

5766 SNA: 0x7A2BCBCFABAA

5767 GPDU: 0x11,0x22,0x33,0x44,0x55,0x66,0x77,0x88,0x99,0x00

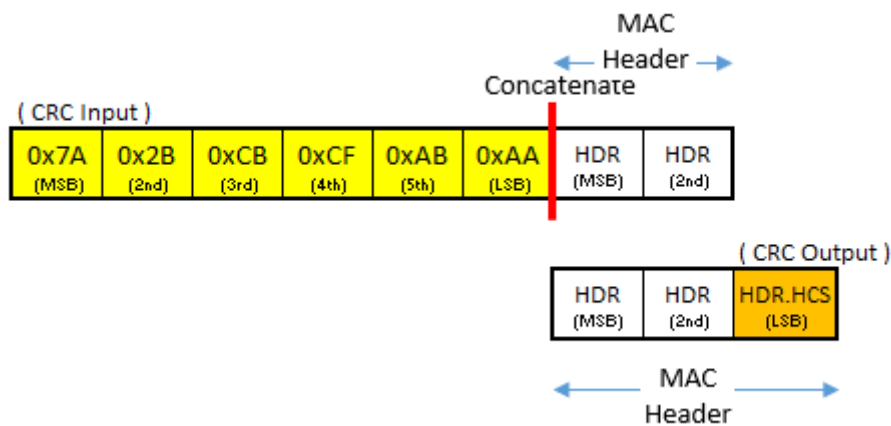
5768 **Table 145 - Example of Concatenation**

|                              |                                                   |
|------------------------------|---------------------------------------------------|
| <b>SNA</b>                   | 0x7A2BCBCFABAA                                    |
| <b>GPDU String</b>           | 0x11,0x22,0x33,0x44,0x55,0x66,0x77,0x88,0x99,0x00 |
| <b>input polynomial M(x)</b> | 0x7A2BCBCFABAA11223344556677889900                |
| <b>Remainder R(x)</b>        | 0xF0EAED6C                                        |
| <b>CRC-32</b>                | 0x3AFA90C8                                        |

5769

5770 The following figures show examples of the CRC inputs for the various PRIME packet types and the relevant  
 5771 chapter in which the frame is described. Each figure contains two parts, the upper part shows the input to  
 5772 the CRC divisor, the lower part shows the frame which is transmitted on the medium.

5773 Figure 154 Shows the generic MAC header as described in section 4.4.2.2

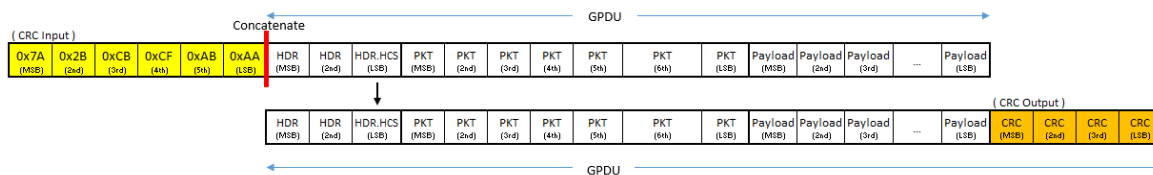


5774

5775 **Figure 154 – Example of CRC in MAC Header**

5776

5777 Figure 155 Shows the generic PDU as described in section 4.4.2.4

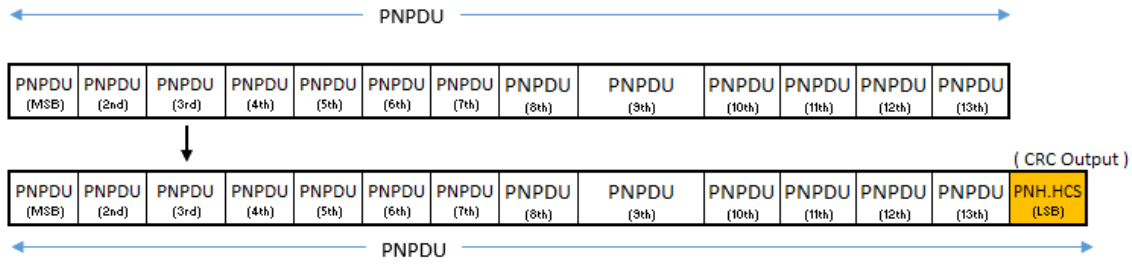


5778

5779 **Figure 155 – Example of CRC in Generic PDU**

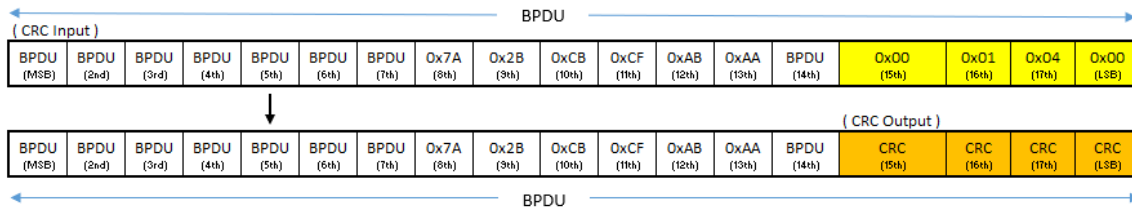
5780

5781 Figure 156 Shows the Promotion Needed PDU as described in section 4.4.3



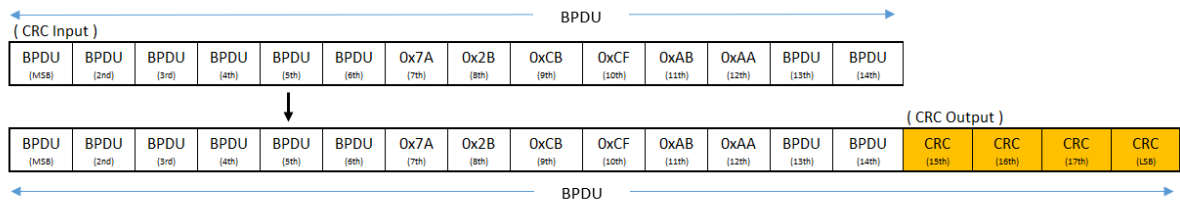
5782  
5783 **Figure 156 – Example of CRC in Promotion Needed PDU**

5784  
5785 Figure 157 Shows the PRIME 1.4 Beacon PDU as described in section 4.4.4



5786  
5787 **Figure 157 – Example of CRC in PRIME 1.4 Beacon PDU**

5788  
5789 Figure 158 Shows the PRIME 1.4 Compatibility Beacon PDU as described in Annex K.1.2



5790  
5791 **Figure 158 – Example of CRC in PRIME 1.4 Compatibility Beacon PDU**

5792

**Annex B**  
**(normative)**  
**EVM calculation**

This annex describes calculation of the EVM by a reference receiver, assuming accurate synchronization and FFT window placement. Let

- $r_k^i$  denotes the FFT output for symbol  $i$  and  $k$  are the indices of data subcarriers.
- $\Delta b_k \in \{0, 1, \dots, P-1\}$  represents the decision on the received information symbol coded in the phase increment.
- $P = 2, 4$ , or  $8$  in the case of DBPSK, DQPSK or D8PSK, respectively.

The EVM definition is then given by;

$$EVM = \frac{\sum_{i=1}^L \sum_{k \in \{\text{data subcarriers}\}} \left( \text{abs}(r_k^i - r_{k-1}^i e^{-(j*2*\pi/P) \times \Delta b_{k-1}}) \right)^2}{\sum_{i=1}^L \sum_{k \in \{\text{data subcarriers}\}} \left( \text{abs}(r_k^i) \right)^2}$$

In the above,  $\text{abs}(\cdot)$  refers to the magnitude of a complex number.  $L$  is the number of OFDM symbols in the most recently received PPDU, over which the EVM is calculated.

The noise can be estimated as the numerator of the EVM. The RSSI can be estimated as the denominator of the EVM. The SNR can be estimated as the reciprocal of the EVM above plus 3dB due to differential decoding.



**Annex C**  
**(informative)**  
**Interleaving matrixes ( $N_{CH} = 1$ )**

**Table 146 - Header interleaving matrix.**

|    |    |    |    |    |    |    |    |    |    |    |    |
|----|----|----|----|----|----|----|----|----|----|----|----|
| 12 | 11 | 10 | 9  | 8  | 7  | 6  | 5  | 4  | 3  | 2  | 1  |
| 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 |
| 36 | 35 | 34 | 33 | 32 | 31 | 30 | 29 | 28 | 27 | 26 | 25 |
| 48 | 47 | 46 | 45 | 44 | 43 | 42 | 41 | 40 | 39 | 38 | 37 |
| 60 | 59 | 58 | 57 | 56 | 55 | 54 | 53 | 52 | 51 | 50 | 49 |
| 72 | 71 | 70 | 69 | 68 | 67 | 66 | 65 | 64 | 63 | 62 | 61 |
| 84 | 83 | 82 | 81 | 80 | 79 | 78 | 77 | 76 | 75 | 74 | 73 |

**Table 147 - DBPSK(FEC ON) interleaving matrix.**

|    |    |    |    |    |    |    |    |    |    |    |    |
|----|----|----|----|----|----|----|----|----|----|----|----|
| 12 | 11 | 10 | 9  | 8  | 7  | 6  | 5  | 4  | 3  | 2  | 1  |
| 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 |
| 36 | 35 | 34 | 33 | 32 | 31 | 30 | 29 | 28 | 27 | 26 | 25 |
| 48 | 47 | 46 | 45 | 44 | 43 | 42 | 41 | 40 | 39 | 38 | 37 |
| 60 | 59 | 58 | 57 | 56 | 55 | 54 | 53 | 52 | 51 | 50 | 49 |
| 72 | 71 | 70 | 69 | 68 | 67 | 66 | 65 | 64 | 63 | 62 | 61 |
| 84 | 83 | 82 | 81 | 80 | 79 | 78 | 77 | 76 | 75 | 74 | 73 |
| 96 | 95 | 94 | 93 | 92 | 91 | 90 | 89 | 88 | 87 | 86 | 85 |

**Table 148 - DQPSK(FEC ON) interleaving matrix.**

|     |     |     |     |     |     |     |     |     |     |     |     |
|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|
| 12  | 11  | 10  | 9   | 8   | 7   | 6   | 5   | 4   | 3   | 2   | 1   |
| 24  | 23  | 22  | 21  | 20  | 19  | 18  | 17  | 16  | 15  | 14  | 13  |
| 36  | 35  | 34  | 33  | 32  | 31  | 30  | 29  | 28  | 27  | 26  | 25  |
| 48  | 47  | 46  | 45  | 44  | 43  | 42  | 41  | 40  | 39  | 38  | 37  |
| 60  | 59  | 58  | 57  | 56  | 55  | 54  | 53  | 52  | 51  | 50  | 49  |
| 72  | 71  | 70  | 69  | 68  | 67  | 66  | 65  | 64  | 63  | 62  | 61  |
| 84  | 83  | 82  | 81  | 80  | 79  | 78  | 77  | 76  | 75  | 74  | 73  |
| 96  | 95  | 94  | 93  | 92  | 91  | 90  | 89  | 88  | 87  | 86  | 85  |
| 108 | 107 | 106 | 105 | 104 | 103 | 102 | 101 | 100 | 99  | 98  | 97  |
| 120 | 119 | 118 | 117 | 116 | 115 | 114 | 113 | 112 | 111 | 110 | 109 |
| 132 | 131 | 130 | 129 | 128 | 127 | 126 | 125 | 124 | 123 | 122 | 121 |
| 144 | 143 | 142 | 141 | 140 | 139 | 138 | 137 | 136 | 135 | 134 | 133 |
| 156 | 155 | 154 | 153 | 152 | 151 | 150 | 149 | 148 | 147 | 146 | 145 |
| 168 | 167 | 166 | 165 | 164 | 163 | 162 | 161 | 160 | 159 | 158 | 157 |
| 180 | 179 | 178 | 177 | 176 | 175 | 174 | 173 | 172 | 171 | 170 | 169 |
| 192 | 191 | 190 | 189 | 188 | 187 | 186 | 185 | 184 | 183 | 182 | 181 |

5817

Table 149 - D8PSK(FEC ON) interleaving matrix.

|     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |
|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|
| 18  | 17  | 16  | 15  | 14  | 13  | 12  | 11  | 10  | 9   | 8   | 7   | 6   | 5   | 4   | 3   | 2   | 1   |
| 36  | 35  | 34  | 33  | 32  | 31  | 30  | 29  | 28  | 27  | 26  | 25  | 24  | 23  | 22  | 21  | 20  | 19  |
| 54  | 53  | 52  | 51  | 50  | 49  | 48  | 47  | 46  | 45  | 44  | 43  | 42  | 41  | 40  | 39  | 38  | 37  |
| 72  | 71  | 70  | 69  | 68  | 67  | 66  | 65  | 64  | 63  | 62  | 61  | 60  | 59  | 58  | 57  | 56  | 55  |
| 90  | 89  | 88  | 87  | 86  | 85  | 84  | 83  | 82  | 81  | 80  | 79  | 78  | 77  | 76  | 75  | 74  | 73  |
| 108 | 107 | 106 | 105 | 104 | 103 | 102 | 101 | 100 | 99  | 98  | 97  | 96  | 95  | 94  | 93  | 92  | 91  |
| 126 | 125 | 124 | 123 | 122 | 121 | 120 | 119 | 118 | 117 | 116 | 115 | 114 | 113 | 112 | 111 | 110 | 109 |
| 144 | 143 | 142 | 141 | 140 | 139 | 138 | 137 | 136 | 135 | 134 | 133 | 132 | 131 | 130 | 129 | 128 | 127 |
| 162 | 161 | 160 | 159 | 158 | 157 | 156 | 155 | 154 | 153 | 152 | 151 | 150 | 149 | 148 | 147 | 146 | 145 |
| 180 | 179 | 178 | 177 | 176 | 175 | 174 | 173 | 172 | 171 | 170 | 169 | 168 | 167 | 166 | 165 | 164 | 163 |
| 198 | 197 | 196 | 195 | 194 | 193 | 192 | 191 | 190 | 189 | 188 | 187 | 186 | 185 | 184 | 183 | 182 | 181 |
| 216 | 215 | 214 | 213 | 212 | 211 | 210 | 209 | 208 | 207 | 206 | 205 | 204 | 203 | 202 | 201 | 200 | 199 |
| 234 | 233 | 232 | 231 | 230 | 229 | 228 | 227 | 226 | 225 | 224 | 223 | 222 | 221 | 220 | 219 | 218 | 217 |
| 252 | 251 | 250 | 249 | 248 | 247 | 246 | 245 | 244 | 243 | 242 | 241 | 240 | 239 | 238 | 237 | 236 | 235 |
| 270 | 269 | 268 | 267 | 266 | 265 | 264 | 263 | 262 | 261 | 260 | 259 | 258 | 257 | 256 | 255 | 254 | 253 |
| 288 | 287 | 286 | 285 | 284 | 283 | 282 | 281 | 280 | 279 | 278 | 277 | 276 | 275 | 274 | 273 | 272 | 271 |

5818

## Annex D (normative) MAC layer constants

This section defines all the MAC layer constants.

**Table 150 - Table of MAC constants**

| Constant                 | Value      | Description                                                                                                                                                                                                                                           |
|--------------------------|------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| MACBeaconLength1         | 5 symbols  | Length of beacon in symbols for PLC single channel bands, Type A frame and DBPSK_CC modulation. This constant has also a special use as it enters in the determination of the minimum CFP (see 4.3.3.1) and the maximum SCP (see 4.3.3.3.1) durations |
| MACBeaconLength2         | 16 symbols | Length of beacon in symbols for PLC single channel bands, Type B frame and DQPSK_R modulation                                                                                                                                                         |
| MACBeaconLength3         | 24 symbols | Length of beacon in symbols for PLC single channel bands, Type B frame and DBPSK_R modulation                                                                                                                                                         |
| MACBeaconLength4         | 19 symbols | Length of beacon in symbols for PLC channel 1, Type BC frame and DQPSK_R modulation                                                                                                                                                                   |
| MACBeaconLength5         | 27 symbols | Length of beacon in symbols for PLC channel 1, Type BC frame and DBPSK_R modulation                                                                                                                                                                   |
| MACBeaconLength6         | 3 symbols  | Length of beacon in symbols for RF band 863–870 MHz, Operating Mode 1, uncoded FSK, with or without channel hopping                                                                                                                                   |
| MACMinSCPLength          | 64 symbols | Minimum length of SCP.                                                                                                                                                                                                                                |
| MACPriorityLevels        | 4          | Number of levels of priority supported by the system.                                                                                                                                                                                                 |
| MACCtrlPktMinRobustness  | DBPSK_CC   | Weakest modulation scheme for Control packets.                                                                                                                                                                                                        |
| MACCtrlPktPriority       | 1          | MAC Priority used to transmit all the Control Packets except ALV Control Packets                                                                                                                                                                      |
| MACALVACKCtrlPktPriority | 0          | MAC Priority used to transmit ALV ACK control Packets                                                                                                                                                                                                 |
| MACSuperFrameLength      | 32         | Number of frames that defines the superframe                                                                                                                                                                                                          |
| MACMaxPRNIgnore          | 3          | Maximum number of Promotion-Needed messages a Terminal can ignore.                                                                                                                                                                                    |

| Constant                    | Value                                                              | Description                                                                                                                       |
|-----------------------------|--------------------------------------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------|
| MACConcurrentAliveProcedure | 2                                                                  | The number of Alive procedure a Service Node shall support at any given time                                                      |
| N <sub>miss-beacon</sub>    | 5                                                                  | Number of superframes a Service Node does not receive an expected beacon before considering its Switch Node as unavailable.       |
| ARQMaxTxCount               | 32                                                                 | Maximum allowed retransmission count before an ARQ connection must be closed                                                      |
| ARQCongClrTime              | 10 sec                                                             | When the receiver has indicated congestion, this time must be waited before retransmitting the data.                              |
| ARQMaxCongInd               | 7                                                                  | After ARQMaxCongInd consecutive transmissions which failed due to congestion, the connection should be declared permanently dead. |
| ARQMaxAckHoldTime           | 7 sec                                                              | Time the receiver may delay sending an ACK in order to allow consolidated ACKs or piggyback the ACK with a data packet.           |
| aUnitBackOffPeriod          | Equal to PHY PIB attributes<br>aTurnaroundTime<br>+ phyCCADuration | The number of symbols forming the basic time period used by the RF CSMA-CA algorithm.                                             |

**Annex E  
(normative)****Convergence layer constants**

The following TYPE values are defined for use by Convergence layers from chapter 5.

**Table 151 - TYPE value assignments**

| TYPE Symbolic Name   | Value |
|----------------------|-------|
| TYPE_CL_IPv4_AR      | 1     |
| TYPE_CL_IPv4_UNICAST | 2     |
| TYPE_CL_432          | 3     |
| TYPE_CL_MGMT         | 4     |
| TYPE_CL_IPv6_AR      | 5     |
| TYPE_CL_IPv6_DATA    | 6     |

The following LCID values apply for broadcast connections defined by Convergence layers from chapter 5.

**Table 152 - LCID value assignments**

| LCID Symbolic Name    | Value | MAC Scope  |
|-----------------------|-------|------------|
| LCI_CL_IPv4_BROADCAST | 1     | Broadcast. |
| LCI_CL_432_BROADCAST  | 2     | Broadcast. |

The following Result values are defined for Convergence layer primitives.

**Table 153 - Result values for Convergence layer primitives**

| Result              | Description                                                                       |
|---------------------|-----------------------------------------------------------------------------------|
| Success = 0         | The SSCS service was successfully performed.                                      |
| Reject = 1          | The SSCS service failed because it was rejected by the base node.                 |
| Timeout = 2         | A timed out occurs during the SSCS service processing                             |
| Not Registered = 6  | The service node is not currently registered to a Subnetwork.                     |
| Unsupported SP = 14 | Device doesn't support SP > 0 but asked for an encrypted connection establishment |

## **Annex F (normative) Profiles**

5834  
5835  
5836

5837 Given the different applications which are foreseen for this specification compliant products, it is necessary  
5838 to define different profiles. Profiles cover the functionalities that represent the respective feature set. They  
5839 need to be implemented as written in order to assure interoperability.

5840 This specification has a number of options, which, if exercised in different ways by different vendors, will  
5841 hamper both compliance testing activities and future product interoperability. The profiles further restrict  
5842 those options so as to promote interoperability and testability.

5843 A specific profile will dictate which capabilities a Node negotiates through the Registering and Promotion  
5844 processes.

### **5845 F.1 Smart Metering Profile**

5846 The following options will be either mandatory or optional for Smart Metering Nodes.

5847 REG.CAP\_SW:

- 5848 • Base Node: Set to 1.
- 5849 • Service Node: Set to 1.

5850 REG.CAP\_PA:

- 5851 • Base Node: optional.
- 5852 • Service Node: optional.

5853 REG.CAP\_CFP:

- 5854 • Base Node: optional.
- 5855 • Service Node: optional.

5856 REG.CAP\_DC

- 5857 • Base Node: optional.
- 5858 • Service Node: optional.

5859 REG.CAP\_MC

- 5860 • Base Node: Set to 1.
- 5861 • Service Node: optional.

5862 REG.CAP\_RM

- 5863 • Base Node: Set to 1.
- 5864 • Service Node: Set to 1.

5865 REG.CAP\_ARQ

- 
- |      |             |                                                                           |
|------|-------------|---------------------------------------------------------------------------|
| 5866 |             | <ul style="list-style-type: none"><li>• Base Node: optional.</li></ul>    |
| 5867 |             | <ul style="list-style-type: none"><li>• Service Node: optional.</li></ul> |
| 5868 | PRO.SWC_DC  |                                                                           |
| 5869 |             | <ul style="list-style-type: none"><li>• Service Node: optional.</li></ul> |
| 5870 | PRO.SWC_MC  |                                                                           |
| 5871 |             | <ul style="list-style-type: none"><li>• Service Node: optional.</li></ul> |
| 5872 | PRO.SWC_RM  |                                                                           |
| 5873 |             | <ul style="list-style-type: none"><li>• Service Node: Set to 1.</li></ul> |
| 5874 | PRO.SWC_ARQ |                                                                           |
| 5875 |             | <ul style="list-style-type: none"><li>• Service Node: optional.</li></ul> |

## Annex G (informative)

### List of frequencies used

The tables below give the exact center frequencies (in Hz) for the 97 subcarriers of the OFDM signal, channel by channel.

Note that a guard period of 15 subcarriers is kept between any two consecutive channels.

**Table 154 – Channel 1: List of frequencies used**

| #   | Frequency   | #   | Frequency   | #   | Frequency   | #   | Frequency   |
|-----|-------------|-----|-------------|-----|-------------|-----|-------------|
| 86  | 41992.18750 | 111 | 54199.21875 | 136 | 66406.25000 | 161 | 78613.28125 |
| 87  | 42480.46875 | 112 | 54687.50000 | 137 | 66894.53125 | 162 | 79101.56250 |
| 88  | 42968.75000 | 113 | 55175.78125 | 138 | 67382.81250 | 163 | 79589.84375 |
| 89  | 43457.03125 | 114 | 55664.06250 | 139 | 67871.09375 | 164 | 80078.12500 |
| 90  | 43945.31250 | 115 | 56152.34375 | 140 | 68359.37500 | 165 | 80566.40625 |
| 91  | 44433.59375 | 116 | 56640.62500 | 141 | 68847.65625 | 166 | 81054.68750 |
| 92  | 44921.87500 | 117 | 57128.90625 | 142 | 69335.93750 | 167 | 81542.96875 |
| 93  | 45410.15625 | 118 | 57617.18750 | 143 | 69824.21875 | 168 | 82031.25000 |
| 94  | 45898.43750 | 119 | 58105.46875 | 144 | 70312.50000 | 169 | 82519.53125 |
| 95  | 46386.71875 | 120 | 58593.75000 | 145 | 70800.78125 | 170 | 83007.81250 |
| 96  | 46875.00000 | 121 | 59082.03125 | 146 | 71289.06250 | 171 | 83496.09375 |
| 97  | 47363.28125 | 122 | 59570.31250 | 147 | 71777.34375 | 172 | 83984.37500 |
| 98  | 47851.56250 | 123 | 60058.59375 | 148 | 72265.62500 | 173 | 84472.65625 |
| 99  | 48339.84375 | 124 | 60546.87500 | 149 | 72753.90625 | 174 | 84960.93750 |
| 100 | 48828.12500 | 125 | 61035.15625 | 150 | 73242.18750 | 175 | 85449.21875 |
| 101 | 49316.40625 | 126 | 61523.43750 | 151 | 73730.46875 | 176 | 85937.50000 |
| 102 | 49804.68750 | 127 | 62011.71875 | 152 | 74218.75000 | 177 | 86425.78125 |
| 103 | 50292.96875 | 128 | 62500.00000 | 153 | 74707.03125 | 178 | 86914.06250 |
| 104 | 50781.25000 | 129 | 62988.28125 | 154 | 75195.31250 | 179 | 87402.34375 |
| 105 | 51269.53125 | 130 | 63476.56250 | 155 | 75683.59375 | 180 | 87890.62500 |



| #   | Frequency   | #   | Frequency   | #   | Frequency   | #   | Frequency   |
|-----|-------------|-----|-------------|-----|-------------|-----|-------------|
| 106 | 51757.81250 | 131 | 63964.84375 | 156 | 76171.87500 | 181 | 88378.90625 |
| 107 | 52246.09375 | 132 | 64453.12500 | 157 | 76660.15625 | 182 | 88867.18750 |
| 108 | 52734.37500 | 133 | 64941.40625 | 158 | 77148.43750 |     |             |
| 109 | 53222.65625 | 134 | 65429.68750 | 159 | 77636.71875 |     |             |
| 110 | 53710.93750 | 135 | 65917.96875 | 160 | 78125.00000 |     |             |

5883

5884

Table 155 – Channel 2: List of frequencies used

| #   | Frequency    | #   | Frequency    | #   | Frequency    | #   | Frequency    |
|-----|--------------|-----|--------------|-----|--------------|-----|--------------|
| 198 | 96679.68750  | 223 | 108886.71875 | 248 | 121093.75000 | 273 | 133300.78125 |
| 199 | 97167.96875  | 224 | 109375.00000 | 249 | 121582.03125 | 274 | 133789.06250 |
| 200 | 97656.25000  | 225 | 109863.28125 | 250 | 122070.31250 | 275 | 134277.34375 |
| 201 | 98144.53125  | 226 | 110351.56250 | 251 | 122558.59375 | 276 | 134765.62500 |
| 202 | 98632.81250  | 227 | 110839.84375 | 252 | 123046.87500 | 277 | 135253.90625 |
| 203 | 99121.09375  | 228 | 111328.12500 | 253 | 123535.15625 | 278 | 135742.18750 |
| 204 | 99609.37500  | 229 | 111816.40625 | 254 | 124023.43750 | 279 | 136230.46875 |
| 205 | 100097.65625 | 230 | 112304.68750 | 255 | 124511.71875 | 280 | 136718.75000 |
| 206 | 100585.93750 | 231 | 112792.96875 | 256 | 125000.00000 | 281 | 137207.03125 |
| 207 | 101074.21875 | 232 | 113281.25000 | 257 | 125488.28125 | 282 | 137695.31250 |
| 208 | 101562.50000 | 233 | 113769.53125 | 258 | 125976.56250 | 283 | 138183.59375 |
| 209 | 102050.78125 | 234 | 114257.81250 | 259 | 126464.84375 | 284 | 138671.87500 |
| 210 | 102539.06250 | 235 | 114746.09375 | 260 | 126953.12500 | 285 | 139160.15625 |
| 211 | 103027.34375 | 236 | 115234.37500 | 261 | 127441.40625 | 286 | 139648.43750 |
| 212 | 103515.62500 | 237 | 115722.65625 | 262 | 127929.68750 | 287 | 140136.71875 |
| 213 | 104003.90625 | 238 | 116210.93750 | 263 | 128417.96875 | 288 | 140625.00000 |
| 214 | 104492.18750 | 239 | 116699.21875 | 264 | 128906.25000 | 289 | 141113.28125 |
| 215 | 104980.46875 | 240 | 117187.50000 | 265 | 129394.53125 | 290 | 141601.56250 |

| #   | Frequency    | #   | Frequency    | #   | Frequency    | #   | Frequency    |
|-----|--------------|-----|--------------|-----|--------------|-----|--------------|
| 216 | 105468.75000 | 241 | 117675.78125 | 266 | 129882.81250 | 291 | 142089.84375 |
| 217 | 105957.03125 | 242 | 118164.06250 | 267 | 130371.09375 | 292 | 142578.12500 |
| 218 | 106445.31250 | 243 | 118652.34375 | 268 | 130859.37500 | 293 | 143066.40625 |
| 219 | 106933.59375 | 244 | 119140.62500 | 269 | 131347.65625 | 294 | 143554.68750 |
| 220 | 107421.87500 | 245 | 119628.90625 | 270 | 131835.93750 |     |              |
| 221 | 107910.15625 | 246 | 120117.18750 | 271 | 132324.21875 |     |              |
| 222 | 108398.43750 | 247 | 120605.46875 | 272 | 132812.50000 |     |              |

5885

5886

Table 156 – Channel 3: List of frequencies used

| #   | Frequency    | #   | Frequency    | #   | Frequency    | #   | Frequency    |
|-----|--------------|-----|--------------|-----|--------------|-----|--------------|
| 310 | 151367.18750 | 335 | 163574.21875 | 360 | 175781.25000 | 385 | 187988.28125 |
| 311 | 151855.46875 | 336 | 164062.50000 | 361 | 176269.53125 | 386 | 188476.56250 |
| 312 | 152343.75000 | 337 | 164550.78125 | 362 | 176757.81250 | 387 | 188964.84375 |
| 313 | 152832.03125 | 338 | 165039.06250 | 363 | 177246.09375 | 388 | 189453.12500 |
| 314 | 153320.31250 | 339 | 165527.34375 | 364 | 177734.37500 | 389 | 189941.40625 |
| 315 | 153808.59375 | 340 | 166015.62500 | 365 | 178222.65625 | 390 | 190429.68750 |
| 316 | 154296.87500 | 341 | 166503.90625 | 366 | 178710.93750 | 391 | 190917.96875 |
| 317 | 154785.15625 | 342 | 166992.18750 | 367 | 179199.21875 | 392 | 191406.25000 |
| 318 | 155273.43750 | 343 | 167480.46875 | 368 | 179687.50000 | 393 | 191894.53125 |
| 319 | 155761.71875 | 344 | 167968.75000 | 369 | 180175.78125 | 394 | 192382.81250 |
| 320 | 156250.00000 | 345 | 168457.03125 | 370 | 180664.06250 | 395 | 192871.09375 |
| 321 | 156738.28125 | 346 | 168945.31250 | 371 | 181152.34375 | 396 | 193359.37500 |
| 322 | 157226.56250 | 347 | 169433.59375 | 372 | 181640.62500 | 397 | 193847.65625 |
| 323 | 157714.84375 | 348 | 169921.87500 | 373 | 182128.90625 | 398 | 194335.93750 |
| 324 | 158203.12500 | 349 | 170410.15625 | 374 | 182617.18750 | 399 | 194824.21875 |
| 325 | 158691.40625 | 350 | 170898.43750 | 375 | 183105.46875 | 400 | 195312.50000 |

| #   | Frequency    | #   | Frequency    | #   | Frequency    | #   | Frequency    |
|-----|--------------|-----|--------------|-----|--------------|-----|--------------|
| 326 | 159179.68750 | 351 | 171386.71875 | 376 | 183593.75000 | 401 | 195800.78125 |
| 327 | 159667.96875 | 352 | 171875.00000 | 377 | 184082.03125 | 402 | 196289.06250 |
| 328 | 160156.25000 | 353 | 172363.28125 | 378 | 184570.31250 | 403 | 196777.34375 |
| 329 | 160644.53125 | 354 | 172851.56250 | 379 | 185058.59375 | 404 | 197265.62500 |
| 330 | 161132.81250 | 355 | 173339.84375 | 380 | 185546.87500 | 405 | 197753.90625 |
| 331 | 161621.09375 | 356 | 173828.12500 | 381 | 186035.15625 | 406 | 198242.18750 |
| 332 | 162109.37500 | 357 | 174316.40625 | 382 | 186523.43750 |     |              |
| 333 | 162597.65625 | 358 | 174804.68750 | 383 | 187011.71875 |     |              |
| 334 | 163085.93750 | 359 | 175292.96875 | 384 | 187500.00000 |     |              |

5887

5888

Table 157 – Channel 4: List of frequencies used

| #   | Frequency    | #   | Frequency    | #   | Frequency    | #   | Frequency    |
|-----|--------------|-----|--------------|-----|--------------|-----|--------------|
| 422 | 206054.68750 | 447 | 218261.71875 | 472 | 230468.75000 | 497 | 242675.78125 |
| 423 | 206542.96875 | 448 | 218750.00000 | 473 | 230957.03125 | 498 | 243164.06250 |
| 424 | 207031.25000 | 449 | 219238.28125 | 474 | 231445.31250 | 499 | 243652.34375 |
| 425 | 207519.53125 | 450 | 219726.56250 | 475 | 231933.59375 | 500 | 244140.62500 |
| 426 | 208007.81250 | 451 | 220214.84375 | 476 | 232421.87500 | 501 | 244628.90625 |
| 427 | 208496.09375 | 452 | 220703.12500 | 477 | 232910.15625 | 502 | 245117.18750 |
| 428 | 208984.37500 | 453 | 221191.40625 | 478 | 233398.43750 | 503 | 245605.46875 |
| 429 | 209472.65625 | 454 | 221679.68750 | 479 | 233886.71875 | 504 | 246093.75000 |
| 430 | 209960.93750 | 455 | 222167.96875 | 480 | 234375.00000 | 505 | 246582.03125 |
| 431 | 210449.21875 | 456 | 222656.25000 | 481 | 234863.28125 | 506 | 247070.31250 |
| 432 | 210937.50000 | 457 | 223144.53125 | 482 | 235351.56250 | 507 | 247558.59375 |
| 433 | 211425.78125 | 458 | 223632.81250 | 483 | 235839.84375 | 508 | 248046.87500 |
| 434 | 211914.06250 | 459 | 224121.09375 | 484 | 236328.12500 | 509 | 248535.15625 |
| 435 | 212402.34375 | 460 | 224609.37500 | 485 | 236816.40625 | 510 | 249023.43750 |

| #   | Frequency    | #   | Frequency    | #   | Frequency    | #   | Frequency    |
|-----|--------------|-----|--------------|-----|--------------|-----|--------------|
| 436 | 212890.62500 | 461 | 225097.65625 | 486 | 237304.68750 | 511 | 249511.71875 |
| 437 | 213378.90625 | 462 | 225585.93750 | 487 | 237792.96875 | 512 | 250000.00000 |
| 438 | 213867.18750 | 463 | 226074.21875 | 488 | 238281.25000 | 513 | 250488.28125 |
| 439 | 214355.46875 | 464 | 226562.50000 | 489 | 238769.53125 | 514 | 250976.56250 |
| 440 | 214843.75000 | 465 | 227050.78125 | 490 | 239257.81250 | 515 | 251464.84375 |
| 441 | 215332.03125 | 466 | 227539.06250 | 491 | 239746.09375 | 516 | 251953.12500 |
| 442 | 215820.31250 | 467 | 228027.34375 | 492 | 240234.37500 | 517 | 252441.40625 |
| 443 | 216308.59375 | 468 | 228515.62500 | 493 | 240722.65625 | 518 | 252929.68750 |
| 444 | 216796.87500 | 469 | 229003.90625 | 494 | 241210.93750 |     |              |
| 445 | 217285.15625 | 470 | 229492.18750 | 495 | 241699.21875 |     |              |
| 446 | 217773.43750 | 471 | 229980.46875 | 496 | 242187.50000 |     |              |

5889

5890

Table 158 – Channel 5: List of frequencies used

| #   | Frequency    | #   | Frequency    | #   | Frequency    | #   | Frequency    |
|-----|--------------|-----|--------------|-----|--------------|-----|--------------|
| 534 | 260742.18750 | 559 | 272949.21875 | 584 | 285156.25000 | 609 | 297363.28125 |
| 535 | 261230.46875 | 560 | 273437.50000 | 585 | 285644.53125 | 610 | 297851.56250 |
| 536 | 261718.75000 | 561 | 273925.78125 | 586 | 286132.81250 | 611 | 298339.84375 |
| 537 | 262207.03125 | 562 | 274414.06250 | 587 | 286621.09375 | 612 | 298828.12500 |
| 538 | 262695.31250 | 563 | 274902.34375 | 588 | 287109.37500 | 613 | 299316.40625 |
| 539 | 263183.59375 | 564 | 275390.62500 | 589 | 287597.65625 | 614 | 299804.68750 |
| 540 | 263671.87500 | 565 | 275878.90625 | 590 | 288085.93750 | 615 | 300292.96875 |
| 541 | 264160.15625 | 566 | 276367.18750 | 591 | 288574.21875 | 616 | 300781.25000 |
| 542 | 264648.43750 | 567 | 276855.46875 | 592 | 289062.50000 | 617 | 301269.53125 |
| 543 | 265136.71875 | 568 | 277343.75000 | 593 | 289550.78125 | 618 | 301757.81250 |
| 544 | 265625.00000 | 569 | 277832.03125 | 594 | 290039.06250 | 619 | 302246.09375 |
| 545 | 266113.28125 | 570 | 278320.31250 | 595 | 290527.34375 | 620 | 302734.37500 |

| #   | Frequency    | #   | Frequency    | #   | Frequency    | #   | Frequency    |
|-----|--------------|-----|--------------|-----|--------------|-----|--------------|
| 546 | 266601.56250 | 571 | 278808.59375 | 596 | 291015.62500 | 621 | 303222.65625 |
| 547 | 267089.84375 | 572 | 279296.87500 | 597 | 291503.90625 | 622 | 303710.93750 |
| 548 | 267578.12500 | 573 | 279785.15625 | 598 | 291992.18750 | 623 | 304199.21875 |
| 549 | 268066.40625 | 574 | 280273.43750 | 599 | 292480.46875 | 624 | 304687.50000 |
| 550 | 268554.68750 | 575 | 280761.71875 | 600 | 292968.75000 | 625 | 305175.78125 |
| 551 | 269042.96875 | 576 | 281250.00000 | 601 | 293457.03125 | 626 | 305664.06250 |
| 552 | 269531.25000 | 577 | 281738.28125 | 602 | 293945.31250 | 627 | 306152.34375 |
| 553 | 270019.53125 | 578 | 282226.56250 | 603 | 294433.59375 | 628 | 306640.62500 |
| 554 | 270507.81250 | 579 | 282714.84375 | 604 | 294921.87500 | 629 | 307128.90625 |
| 555 | 270996.09375 | 580 | 283203.12500 | 605 | 295410.15625 | 630 | 307617.18750 |
| 556 | 271484.37500 | 581 | 283691.40625 | 606 | 295898.43750 |     |              |
| 557 | 271972.65625 | 582 | 284179.68750 | 607 | 296386.71875 |     |              |
| 558 | 272460.93750 | 583 | 284667.96875 | 608 | 296875.00000 |     |              |

5891

5892

Table 159 – Channel 6: List of frequencies used

| #   | Frequency    | #   | Frequency    | #   | Frequency    | #   | Frequency    |
|-----|--------------|-----|--------------|-----|--------------|-----|--------------|
| 646 | 315429.68750 | 671 | 327636.71875 | 696 | 339843.75000 | 721 | 352050.78125 |
| 647 | 315917.96875 | 672 | 328125.00000 | 697 | 340332.03125 | 722 | 352539.06250 |
| 648 | 316406.25000 | 673 | 328613.28125 | 698 | 340820.31250 | 723 | 353027.34375 |
| 649 | 316894.53125 | 674 | 329101.56250 | 699 | 341308.59375 | 724 | 353515.62500 |
| 650 | 317382.81250 | 675 | 329589.84375 | 700 | 341796.87500 | 725 | 354003.90625 |
| 651 | 317871.09375 | 676 | 330078.12500 | 701 | 342285.15625 | 726 | 354492.18750 |
| 652 | 318359.37500 | 677 | 330566.40625 | 702 | 342773.43750 | 727 | 354980.46875 |
| 653 | 318847.65625 | 678 | 331054.68750 | 703 | 343261.71875 | 728 | 355468.75000 |
| 654 | 319335.93750 | 679 | 331542.96875 | 704 | 343750.00000 | 729 | 355957.03125 |
| 655 | 319824.21875 | 680 | 332031.25000 | 705 | 344238.28125 | 730 | 356445.31250 |

| #   | Frequency    | #   | Frequency    | #   | Frequency    | #   | Frequency    |
|-----|--------------|-----|--------------|-----|--------------|-----|--------------|
| 656 | 320312.50000 | 681 | 332519.53125 | 706 | 344726.56250 | 731 | 356933.59375 |
| 657 | 320800.78125 | 682 | 333007.81250 | 707 | 345214.84375 | 732 | 357421.87500 |
| 658 | 321289.06250 | 683 | 333496.09375 | 708 | 345703.12500 | 733 | 357910.15625 |
| 659 | 321777.34375 | 684 | 333984.37500 | 709 | 346191.40625 | 734 | 358398.43750 |
| 660 | 322265.62500 | 685 | 334472.65625 | 710 | 346679.68750 | 735 | 358886.71875 |
| 661 | 322753.90625 | 686 | 334960.93750 | 711 | 347167.96875 | 736 | 359375.00000 |
| 662 | 323242.18750 | 687 | 335449.21875 | 712 | 347656.25000 | 737 | 359863.28125 |
| 663 | 323730.46875 | 688 | 335937.50000 | 713 | 348144.53125 | 738 | 360351.56250 |
| 664 | 324218.75000 | 689 | 336425.78125 | 714 | 348632.81250 | 739 | 360839.84375 |
| 665 | 324707.03125 | 690 | 336914.06250 | 715 | 349121.09375 | 740 | 361328.12500 |
| 666 | 325195.31250 | 691 | 337402.34375 | 716 | 349609.37500 | 741 | 361816.40625 |
| 667 | 325683.59375 | 692 | 337890.62500 | 717 | 350097.65625 | 742 | 362304.68750 |
| 668 | 326171.87500 | 693 | 338378.90625 | 718 | 350585.93750 |     |              |
| 669 | 326660.15625 | 694 | 338867.18750 | 719 | 351074.21875 |     |              |
| 670 | 327148.43750 | 695 | 339355.46875 | 720 | 351562.50000 |     |              |

5893

5894

Table 160 – Channel 7: List of frequencies used

| #   | Frequency    | #   | Frequency    | #   | Frequency    | #   | Frequency    |
|-----|--------------|-----|--------------|-----|--------------|-----|--------------|
| 758 | 370117.18750 | 783 | 382324.21875 | 808 | 394531.25000 | 833 | 406738.28125 |
| 759 | 370605.46875 | 784 | 382812.50000 | 809 | 395019.53125 | 834 | 407226.56250 |
| 760 | 371093.75000 | 785 | 383300.78125 | 810 | 395507.81250 | 835 | 407714.84375 |
| 761 | 371582.03125 | 786 | 383789.06250 | 811 | 395996.09375 | 836 | 408203.12500 |
| 762 | 372070.31250 | 787 | 384277.34375 | 812 | 396484.37500 | 837 | 408691.40625 |
| 763 | 372558.59375 | 788 | 384765.62500 | 813 | 396972.65625 | 838 | 409179.68750 |
| 764 | 373046.87500 | 789 | 385253.90625 | 814 | 397460.93750 | 839 | 409667.96875 |
| 765 | 373535.15625 | 790 | 385742.18750 | 815 | 397949.21875 | 840 | 410156.25000 |

| #   | Frequency    | #   | Frequency    | #   | Frequency    | #   | Frequency    |
|-----|--------------|-----|--------------|-----|--------------|-----|--------------|
| 766 | 374023.43750 | 791 | 386230.46875 | 816 | 398437.50000 | 841 | 410644.53125 |
| 767 | 374511.71875 | 792 | 386718.75000 | 817 | 398925.78125 | 842 | 411132.81250 |
| 768 | 375000.00000 | 793 | 387207.03125 | 818 | 399414.06250 | 843 | 411621.09375 |
| 769 | 375488.28125 | 794 | 387695.31250 | 819 | 399902.34375 | 844 | 412109.37500 |
| 770 | 375976.56250 | 795 | 388183.59375 | 820 | 400390.62500 | 845 | 412597.65625 |
| 771 | 376464.84375 | 796 | 388671.87500 | 821 | 400878.90625 | 846 | 413085.93750 |
| 772 | 376953.12500 | 797 | 389160.15625 | 822 | 401367.18750 | 847 | 413574.21875 |
| 773 | 377441.40625 | 798 | 389648.43750 | 823 | 401855.46875 | 848 | 414062.50000 |
| 774 | 377929.68750 | 799 | 390136.71875 | 824 | 402343.75000 | 849 | 414550.78125 |
| 775 | 378417.96875 | 800 | 390625.00000 | 825 | 402832.03125 | 850 | 415039.06250 |
| 776 | 378906.25000 | 801 | 391113.28125 | 826 | 403320.31250 | 851 | 415527.34375 |
| 777 | 379394.53125 | 802 | 391601.56250 | 827 | 403808.59375 | 852 | 416015.62500 |
| 778 | 379882.81250 | 803 | 392089.84375 | 828 | 404296.87500 | 853 | 416503.90625 |
| 779 | 380371.09375 | 804 | 392578.12500 | 829 | 404785.15625 | 854 | 416992.18750 |
| 780 | 380859.37500 | 805 | 393066.40625 | 830 | 405273.43750 |     |              |
| 781 | 381347.65625 | 806 | 393554.68750 | 831 | 405761.71875 |     |              |
| 782 | 381835.93750 | 807 | 394042.96875 | 832 | 406250.00000 |     |              |

5895

5896

Table 161 – Channel 8: List of frequencies used

| #   | Frequency    | #   | Frequency    | #   | Frequency    | #   | Frequency    |
|-----|--------------|-----|--------------|-----|--------------|-----|--------------|
| 870 | 424804.68750 | 895 | 437011.71875 | 920 | 449218.75000 | 945 | 461425.78125 |
| 871 | 425292.96875 | 896 | 437500.00000 | 921 | 449707.03125 | 946 | 461914.06250 |
| 872 | 425781.25000 | 897 | 437988.28125 | 922 | 450195.31250 | 947 | 462402.34375 |
| 873 | 426269.53125 | 898 | 438476.56250 | 923 | 450683.59375 | 948 | 462890.62500 |
| 874 | 426757.81250 | 899 | 438964.84375 | 924 | 451171.87500 | 949 | 463378.90625 |
| 875 | 427246.09375 | 900 | 439453.12500 | 925 | 451660.15625 | 950 | 463867.18750 |

| #   | Frequency    | #   | Frequency    | #   | Frequency    | #   | Frequency    |
|-----|--------------|-----|--------------|-----|--------------|-----|--------------|
| 876 | 427734.37500 | 901 | 439941.40625 | 926 | 452148.43750 | 951 | 464355.46875 |
| 877 | 428222.65625 | 902 | 440429.68750 | 927 | 452636.71875 | 952 | 464843.75000 |
| 878 | 428710.93750 | 903 | 440917.96875 | 928 | 453125.00000 | 953 | 465332.03125 |
| 879 | 429199.21875 | 904 | 441406.25000 | 929 | 453613.28125 | 954 | 465820.31250 |
| 880 | 429687.50000 | 905 | 441894.53125 | 930 | 454101.56250 | 955 | 466308.59375 |
| 881 | 430175.78125 | 906 | 442382.81250 | 931 | 454589.84375 | 956 | 466796.87500 |
| 882 | 430664.06250 | 907 | 442871.09375 | 932 | 455078.12500 | 957 | 467285.15625 |
| 883 | 431152.34375 | 908 | 443359.37500 | 933 | 455566.40625 | 958 | 467773.43750 |
| 884 | 431640.62500 | 909 | 443847.65625 | 934 | 456054.68750 | 959 | 468261.71875 |
| 885 | 432128.90625 | 910 | 444335.93750 | 935 | 456542.96875 | 960 | 468750.00000 |
| 886 | 432617.18750 | 911 | 444824.21875 | 936 | 457031.25000 | 961 | 469238.28125 |
| 887 | 433105.46875 | 912 | 445312.50000 | 937 | 457519.53125 | 962 | 469726.56250 |
| 888 | 433593.75000 | 913 | 445800.78125 | 938 | 458007.81250 | 963 | 470214.84375 |
| 889 | 434082.03125 | 914 | 446289.06250 | 939 | 458496.09375 | 964 | 470703.12500 |
| 890 | 434570.31250 | 915 | 446777.34375 | 940 | 458984.37500 | 965 | 471191.40625 |
| 891 | 435058.59375 | 916 | 447265.62500 | 941 | 459472.65625 | 966 | 471679.68750 |
| 892 | 435546.87500 | 917 | 447753.90625 | 942 | 459960.93750 |     |              |
| 893 | 436035.15625 | 918 | 448242.18750 | 943 | 460449.21875 |     |              |
| 894 | 436523.43750 | 919 | 448730.46875 | 944 | 460937.50000 |     |              |



**Annex H**  
**(informative)**  
**Informative**

5897  
5898  
5899

5900 **H.1 Data exchange between to IP communication peers**

5901 This example shows the primitive exchange between a service node (192.168.0.100/24) and a base node  
5902 when the former wants to exchange IP packets with a third service node (192.168.0.101/24) whose IP address  
5903 is in the same IP Subnetwork.

5904 This example makes the following assumptions:

- 5905           • Service node (192.168.0.100) IPv4 SSCS does not exist so it needs to start a IPv4 SSCS and register  
5906           its IP address in the base node prior to the exchange of IP packets.  
5907           • Service node (192.168.0.101) has already registered its IP Address in the base node.

5908 The steps illustrated in next page are:

5909 1. The IPv4 layer of the service node (192.168.0.100) invokes the CL\_IPv4\_ESTABLISH.request primitive. To  
5910 establish IPv4 SSCS, it is required,

5911           a. To establish a connection with the base node so all address resolution messages can be exchanged  
5912           over it.

5913           b. To inform the service node MAC layer that IPv4 SSCS is ready to receive all IPv4 broadcasts packets.  
5914           Note the difference between broadcast and multicast. To join a multicast group, the service node will  
5915           need to inform the base node of the group it wants to join. This is illustrated in section A.2

5916 2. The IPv4\_ layer, once the IPv4 SSCS is established, needs to register its IP address in the base node. To do  
5917 so, it will use the already established connection.

5918 3. Whenever the IPv4\_ needs to deliver an IPv4 packet to a new destination IP address, the following two  
5919 steps are to be done (in this example, the destination IP address is 192.168.0.101).

5920           a. As the IPv4 destination address is new, the IPv4 SSCS needs to request the EUI-48 associated to that  
5921           IPv4 address. To do so, a lookup request message is sent to the base node.

5922           b. Upon the reception of the EUI-48, a new connection (type = TYPE\_CL\_IPv4\_UNICAST) is established  
5923           so that all IP packets to be exchanged between 192.168.0.100 and 192.168.0.101 will use that  
5924           connection.

5925

5926

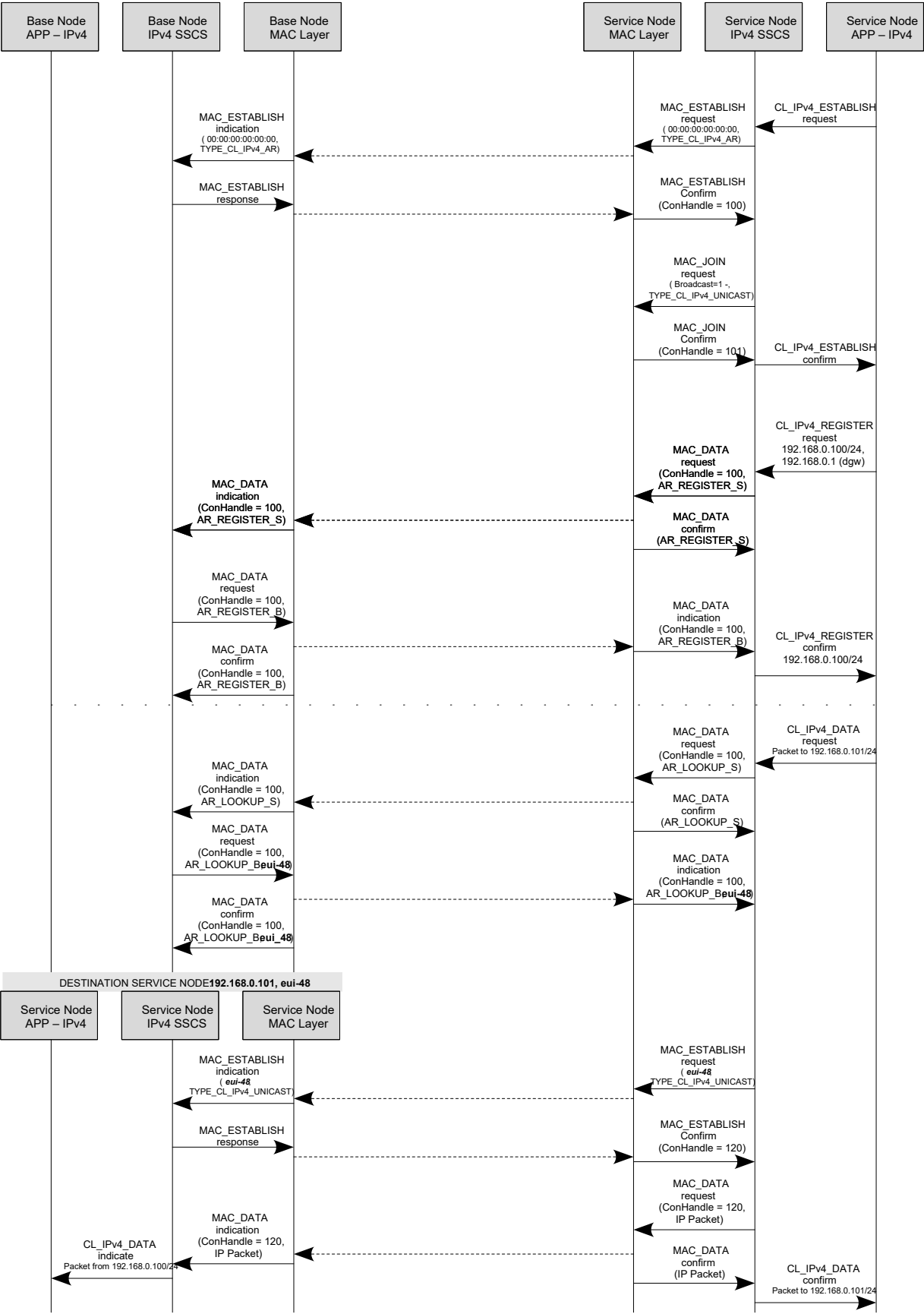


Figure 159 - MSC of IPv4 SCS services

H.2 Joining a multicast group

The figure below illustrates how a service node joins a multicast group. As mentioned before, main difference between multicast and broadcast is related to the messages exchanged. For broadcast, the MAC layer will immediately issue a MAC\_JOIN.confirm primitive since it does not need to perform any end-to-end operation. For multicast, the MAC\_JOIN.confirm is only sent once the Control Packet transaction between the service node and base node is complete.

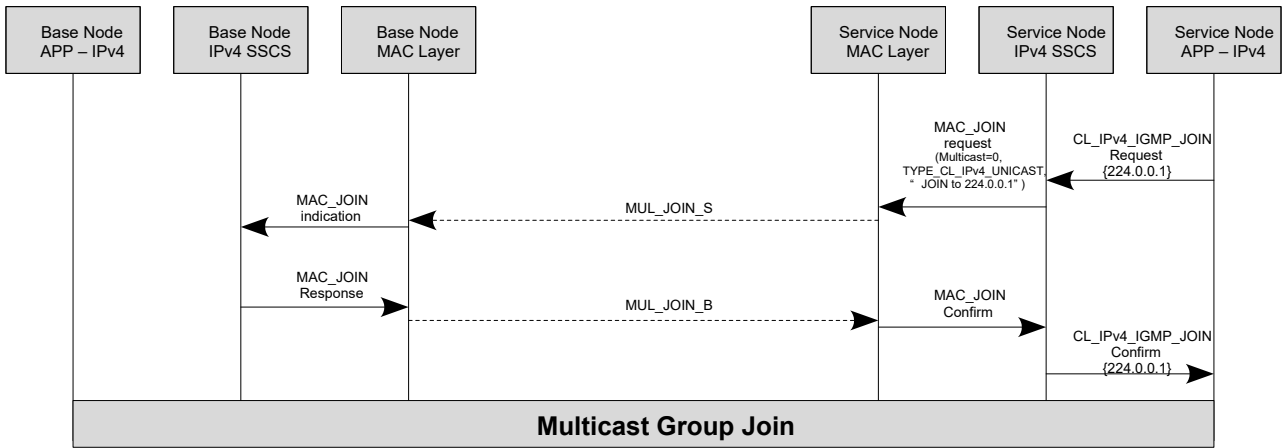


Figure 160 - Joining MS

5938 The MSC below shows the 432 connection establishment and release. 432 SSCS is connection oriented.  
5939 Before any 432\_Data service can take place a connection establishment has to take place. The service node  
5940 upper layer request a connexion establishment to thez 432 SSCS by providing to it the device identifier as  
5941 parameter for the CL\_432\_Establish.request. With the help od the MAC layer services, the service node 432  
5942 SSCS request a connection establishment to the base node. This last one when the connection establishment  
5943 is successful, notifies to the upper layers that a service node has joined the network with the help of the  
5944 CL\_432\_Join.indication primitive and provides to the concerned service node a SSCS destination address in  
5945 addition to its own SSCS address with the help of the MAC\_Establish.response which crries out these  
5946 parameters.

5947 The CL\_432\_release service ends the connection. It is requested by the service node upper layer to the 432  
5948 SSCS which perform it with the help of MAC layer primitives. At the base node side the 432 SSCS notifies the  
5949 end of the connection to the upper layer by a CL\_432\_Leave.indication.

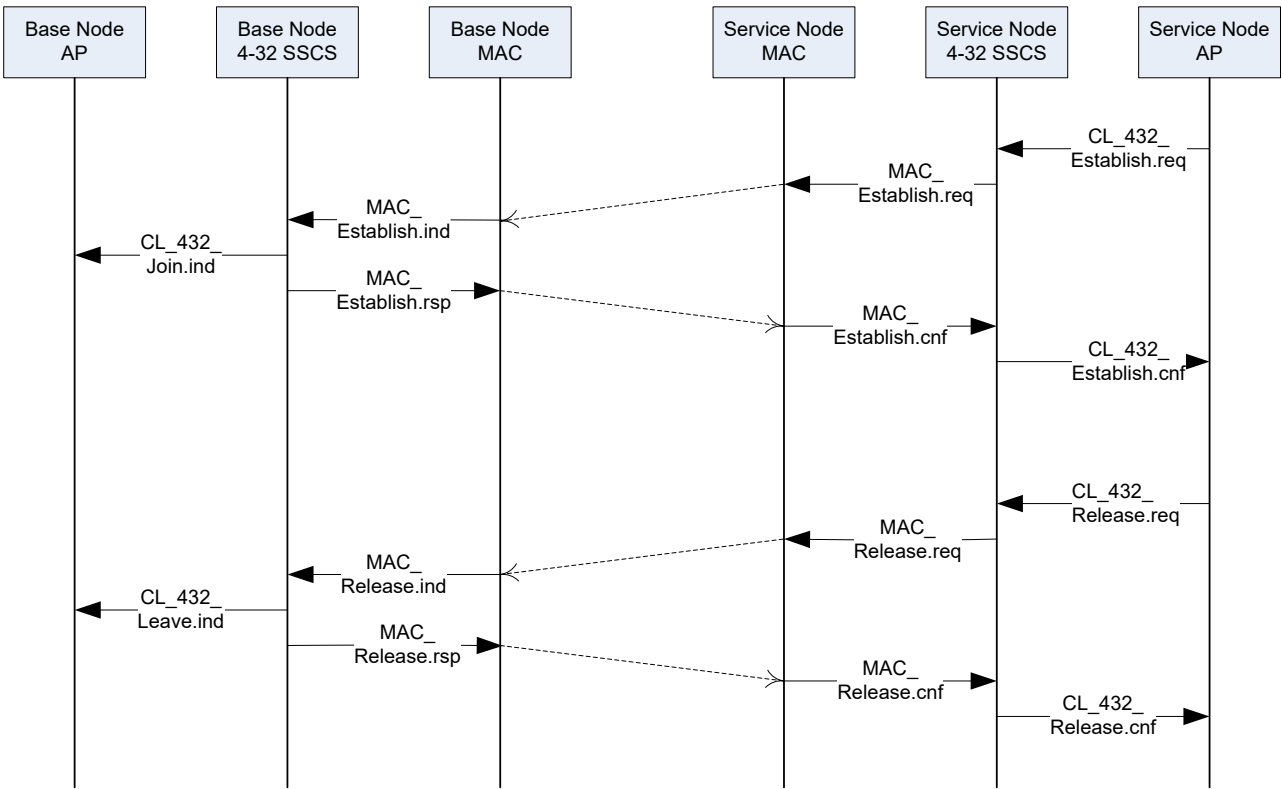


Figure 161 - MSC 432 SSCS services

**Annex I**  
**(informative)**  
**ARQ algorithm**

The algorithm described here is just a recommendation with good performance and aims to better describe how ARQ works. However manufacturers could use a different algorithm as long as it complies with the specification.

When a packet is received the packet ID should be checked. If it is the expected ID and contains data, it shall be processed normally. If the packet does not contain data, it can be discarded. If the ID does not match with the one expected, it is from the future and fits in the input window, then for all the packets not received with ID from the last one received to this one, we can assume that they are lost. If the packet contains data, save that data to pass it to the CL once all the packets before have been received and processed by CL.

If the packet ID does not fit in the input window, we can assume that it is a retransmission that has been delayed, and may be ignored.

If there is any NACK all the packets with PKTID lower than the first NACK in the list have been correctly received, and they can be removed from the transmitting window. If there is not any NACK and there is an ACK, the packets before the received ACK have been received and can be removed from the transmission window. All the packets in the NACK list should be retransmitted as soon as possible.

These are some situations for the transmitter to set the flush bit that may improve the average performance:

- When the window of either the transmitter or the receiver is filled;
- When explicitly requested by the CL;
- After a period of time as a timeout.

The receiver has no responsibility over the ACK send process other than sending them when the transmitter sets the flush bit. Although it has some control over the flow control by the window field. On the other hand the receiver is able to send an ACK if it improves the ARQ performance in a given scenario. One example of this, applicable in most cases, could be making the receiver send an ACK if a period of time has been passed since the last sent ACK, to improve the bandwidth usage (and omit the timeout flush in the transmitter). In those situations the transmitter still has the responsibility to interoperate with the simplest receiver (that does not send it by itself).

It is recommended that the ARQ packet sender maintains a timer for every unacknowledged packet. If the packet cannot get successfully acknowledged when the timer expires, the packet will be retransmitted. This kind of timeout retry works independently with the NACK-initiated retries. After a pre-defined maximum number of timeout retries, it is strongly recommended to tear down the connection. This timeout and connection-teardown mechanism is to prevent the Node retry the ARQ packet forever. The exact number of the timeout values and the timeout retries are left for vendor's own choice.

Annex J

(normative)

PHY backwards compatibility mechanism with PRIME v1.3.6

PRIME specification version V1.4 is an extension of version V1.3.6. The inclusion of new features, such as additional robust modes and a new frame type (Type B), implies that PRIME v1.4 compliant devices shall be able to support the following scenarios:

- Homogeneous networks which do not implement neither the new frame type (Type B) defined in Section 3.3.2 nor the additional robust modes (Robust DBPSK, Robust DQPSK).
- Homogeneous networks which implement the new frame type (Type B) defined in Section 3.3.2 as well as the additional robust modes (Robust DBPSK, Robust DQPSK).
- Mixed networks, composed of a combination of devices described in points (1) and (2) above.

Cases (1) and (2) are trivial since the networks are homogeneous and all devices implement the same features. However, case (3) “Mixed networks” requires a specific mechanism that provides compatibility between PRIME compliant devices using different feature sets. Please note that compatibility between case (1) and case (2) devices could be trivially achieved forcing those devices with an extended set of features to ignore them and to use a more limited configuration (e.g., frame Type A and no robust modes). Nonetheless, the aim of this Annex is to define a backwards compatibility mechanism for mixed networks that allows devices with different feature sets to be part of the same network.

Taking the PHY frame types into account, a backwards compatible frame (“BC frame”) is defined, as shown in Figure 162:

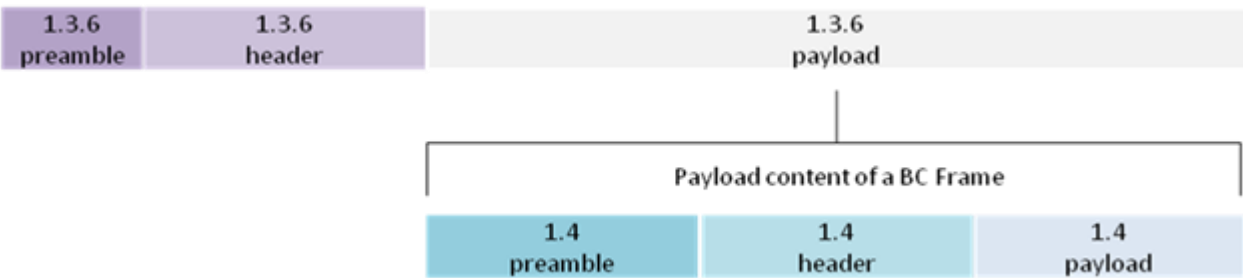
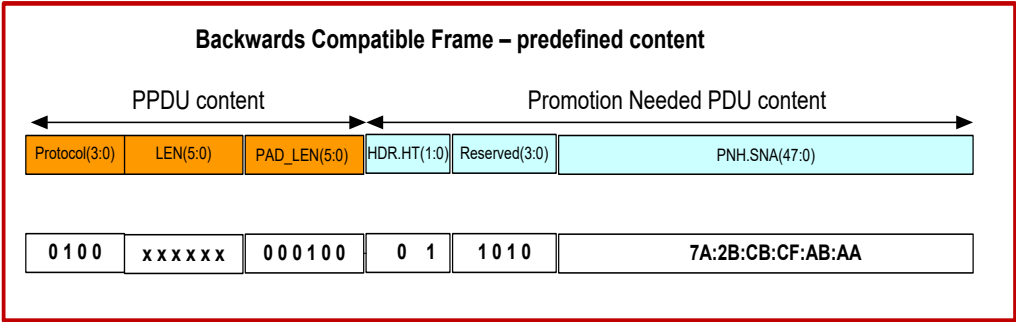


Figure 162 - Backwards Compatible PHY frame

The BC frame is compatible with PRIME v1.3.6 frame at PHY level, since it is just a v1.3.6 PPDU (corresponding to a v1.4 Type A PPDU) encapsulating a v1.4 Type B PPDU.

- BC frame predefined content is described below in Figure 163:
- PPDU content
    - Protocol [3:0]: Default transmission scheme, equal to DBPSK\_CC
    - LEN[5:0]: Type A payload length (number of symbols in Type B header and Type B payload + 4)
  - PNPDU content

- 6020
- 6021
- 6022
- HDR.HT[1:0]: default PNPDU value, equal to “1”
  - Reserved[3:0]: predefined sequence, equal to “1010”
  - PNH.SNA[47:0]: predefined value, "7A:2B:CB:CF:AB:AA"



4.4.2 Promotion Needed PDU

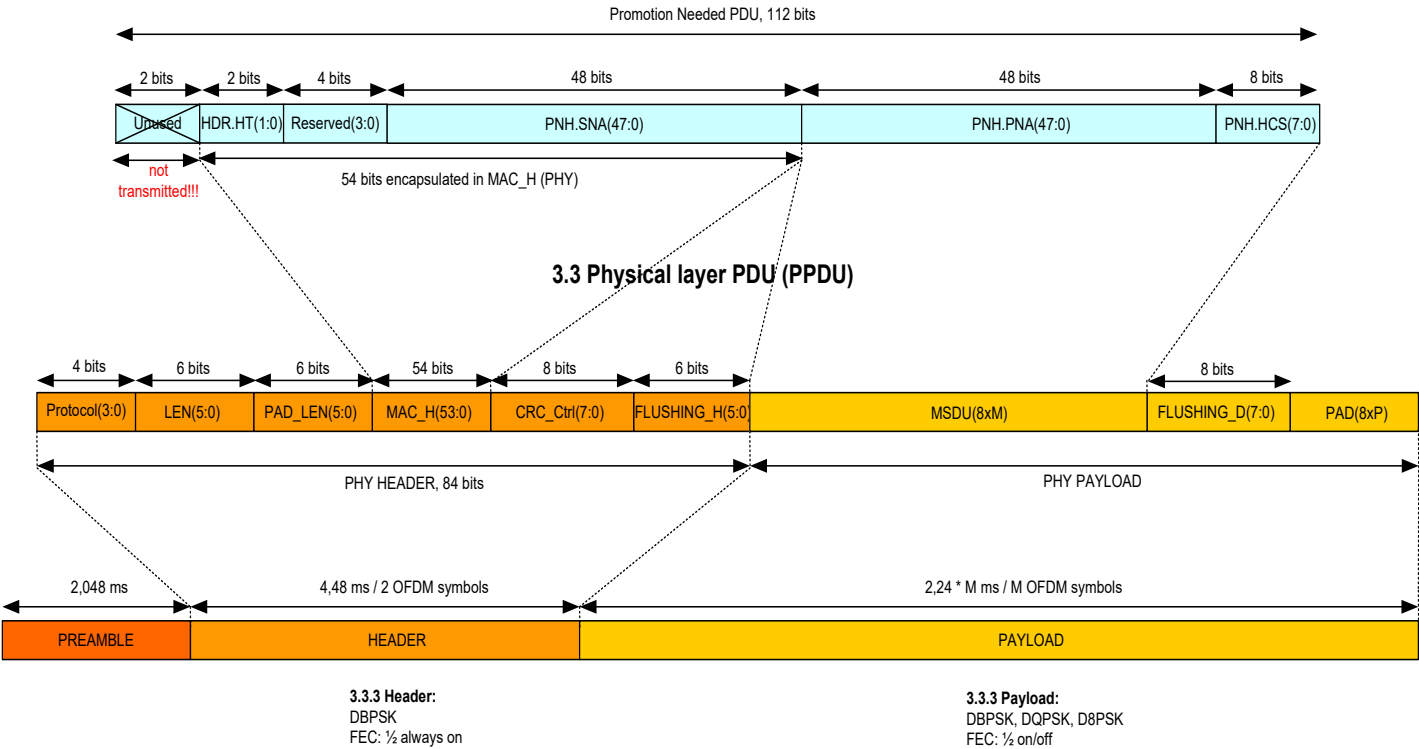


Figure 163 - BC frame predefined content

In mixed networks, the behavior of v1.4 and v1.3.6 devices upon reception of a BC frame will be different:

- v1.3.6 devices detect preamble and header. The content of the v1.3.6 header in a BC frame is a predefined value (see Figure 163). The MAC of v1.3.6 devices will automatically discard the BC frame, but it will not provoke any collisions while the frame is being transmitted (Figure 164).

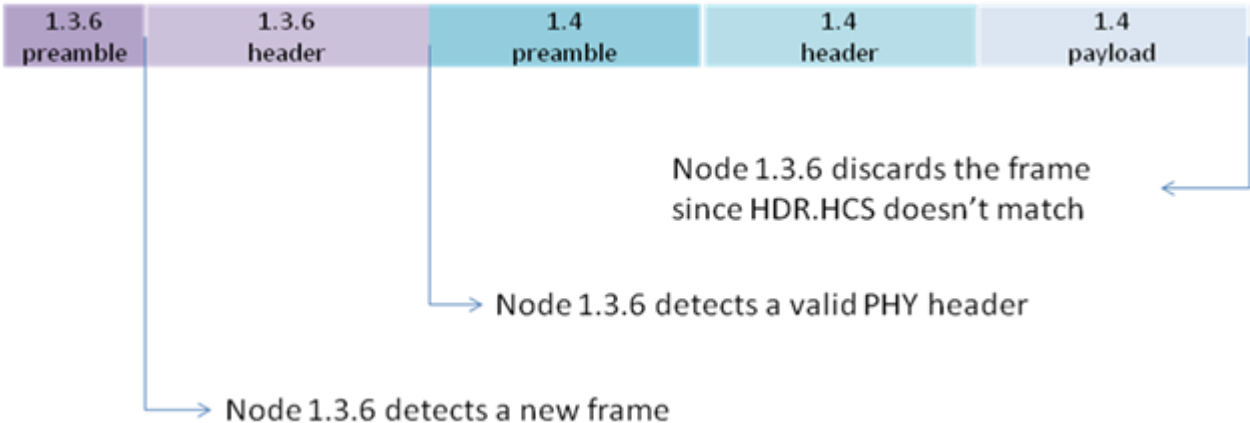


Figure 164 - PHY BC frame detected by v1.3.6 devices

2. 1.4 devices will detect the v1.3.6 preamble and immediately after that the predefined MAC\_H configuration described above. Consequently, a v1.4 device will identify that a BC frame has been received and shall start searching the v1.4 preamble and header (Figure 165):

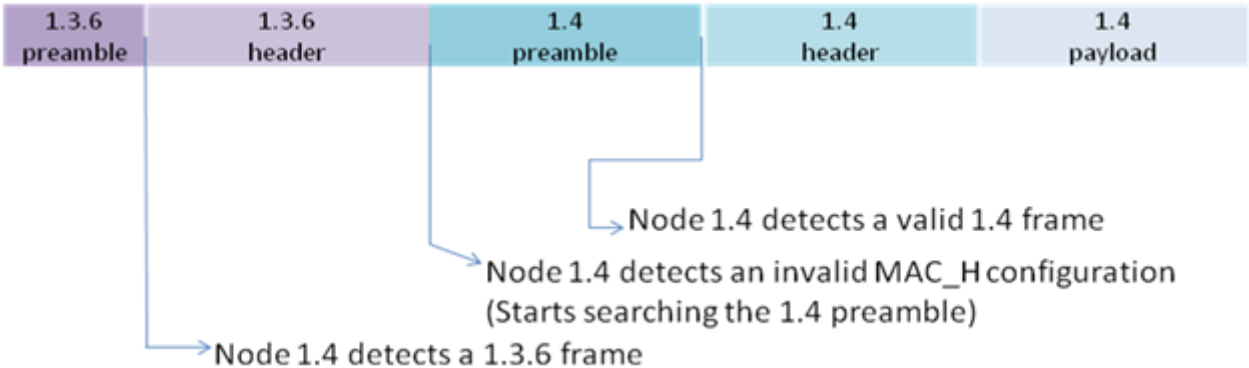


Figure 165 - BC PHY frame detected by v1.4 devices

Two additional use cases have been added to this Annex for the sake of clarification:

1. 1.3.6 frame received by a v1.4 node (Figure 166):

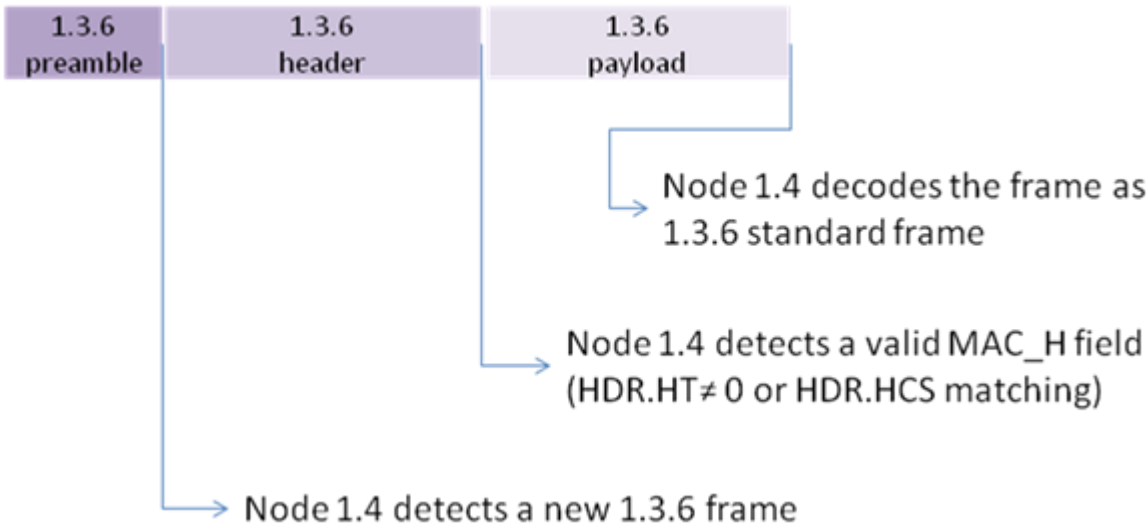




Figure 166 - v1.3.6 frame received by a v1.4 node

2. BC frame received by a v1.4 node in a very hard environment (Figure 167):

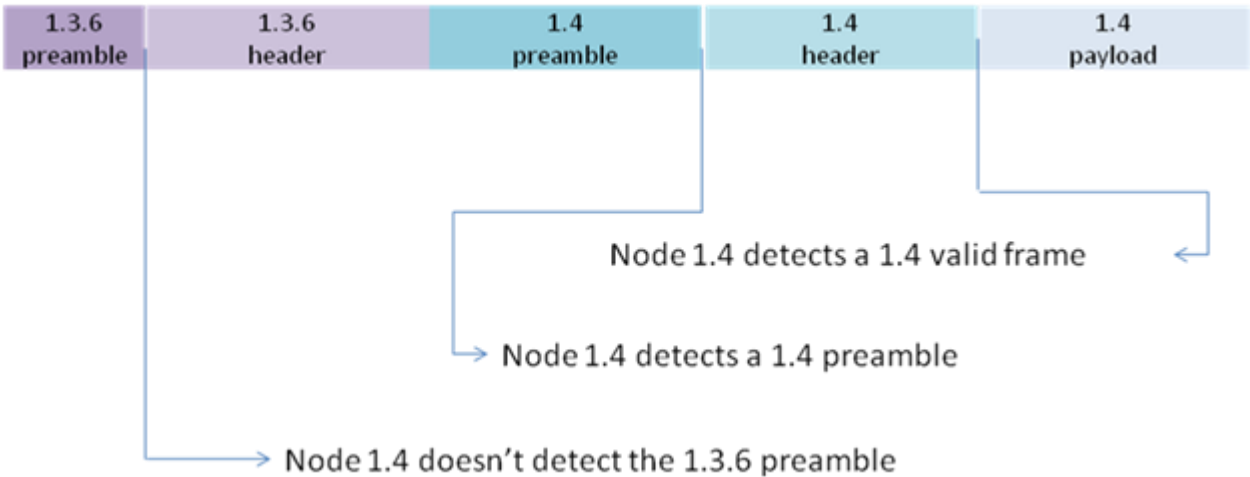


Figure 167 - BC PHY frame in noisy environment

## Annex K (normative)

### MAC Backward Compatibility PDUs and Procedures

#### K.1 MAC PDU format

##### K.1.1 Generic MAC PDU

In a network running in PRIME compatibility mode, all nodes shall use the standard Generic Mac header, as enumerated in Section 4.4.2.2, and the compatibility packet header (CPKT). The compatibility packet header is 6 bytes in length and its composition is shown in Figure 168. Table 162 enumerates the description of each field.

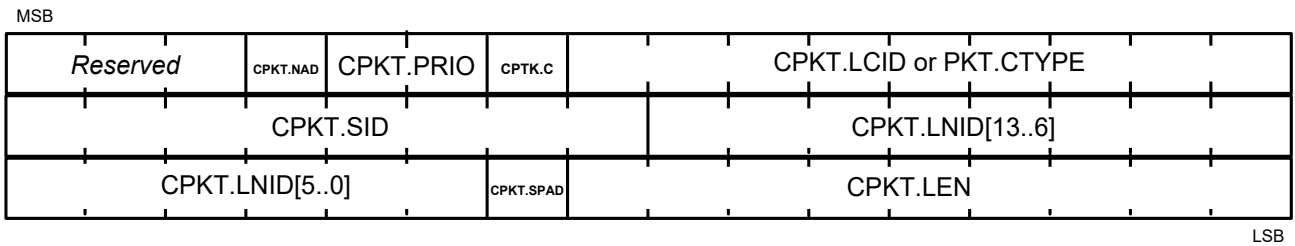


Figure 168 – Compatibility Packet Header

Table 162 – Compatibility packet header fields

| Name                      | Length | Description                                                                                                                                                                                                                                                 |
|---------------------------|--------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Reserved                  | 3 bits | Always 0 for this version of the specification. Reserved for future use.                                                                                                                                                                                    |
| CPKT.NAD                  | 1 bit  | No Aggregation at Destination <ul style="list-style-type: none"> <li>If CPKT.NAD=0 the packet may be aggregated with other packets at destination.</li> <li>If CPKT.NAD=1 the packet may not be aggregated with other packets at destination.</li> </ul>    |
| CPKT.PRIO                 | 2 bits | Indicates packet priority between 0 and 3.                                                                                                                                                                                                                  |
| CPKT.C                    | 1 bits | Control <ul style="list-style-type: none"> <li>If CPKT.C=0 it is a data packet.</li> <li>If CPKT.C=1 it is a control packet.</li> </ul>                                                                                                                     |
| CPKT.LCID /<br>CPKT.CTYPE | 9 bits | Local Connection Identifier or Control Type <ul style="list-style-type: none"> <li>If CPKT.C=0, CPKT.LCID represents the Local Connection Identifier of data packet.</li> <li>If CPKT.C=1, CPKT.CTYPE represents the type of the control packet.</li> </ul> |

| Name      | Length  | Description                                                                                                                                                                                                                 |
|-----------|---------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| CPKT.SID  | 8 bits  | Switch identifier <ul style="list-style-type: none"> <li>• If HDR.DO=0, CPKT.SID represents the SID of the packet source.</li> <li>• If HDR.DO=1, CPKT.SID represents the SID of the packet destination.</li> </ul>         |
| CPKT.LNID | 14 bits | Local Node identifier. <ul style="list-style-type: none"> <li>• If HDR.DO=0, CPKT.LNID represents the LNID of the packet source</li> <li>• If HDR.DO=1, CPKT.LNID represents the LNID of the packet destination.</li> </ul> |
| CPKT.SPAD | 1bit    | Indicates if padding is inserted while encrypting payload. Note that this bit is only of relevance when Security Profile 1 (see 4.3.8.2.2) is used.                                                                         |
| CPKT.LEN  | 9 bits  | Length of the packet payload in bytes.                                                                                                                                                                                      |

6058

6059 **K.1.1.1 MAC control packets**

6060 The CPKT.CTYPE field follows the same enumeration as the PKT.CTYPE field (see Table 21). Control packet  
6061 retransmission shall follow the mechanisms described in Section 4.4.2.6.2.

6062 **K.1.1.1.1 Compatibility REG control packet (CREG, CPKT.CTYPE=1)**

6063 The CREG control packet shall be used for registration requests (REG\_REQ) in any case.

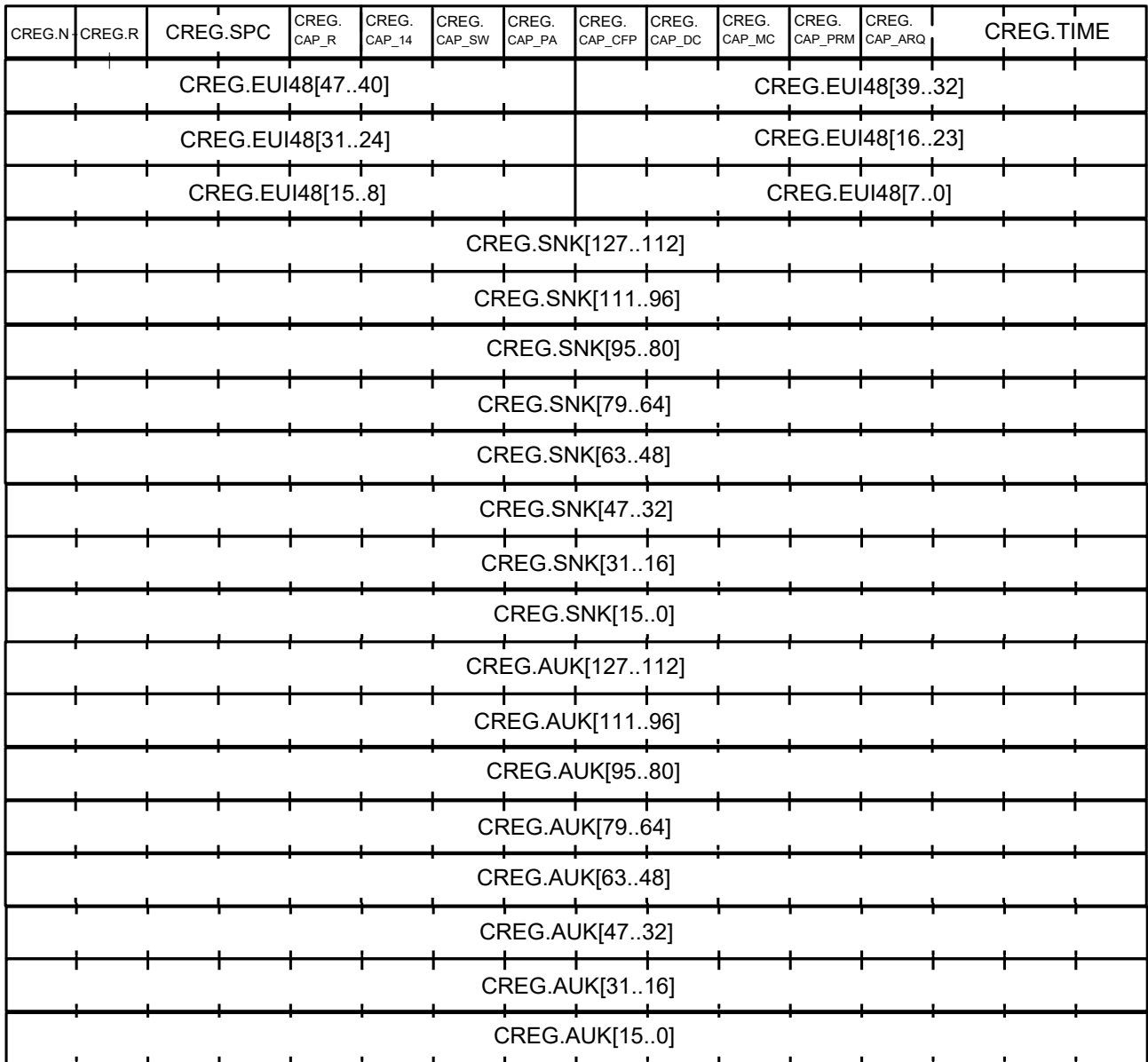
6064 REG and CREG control packets are distinguished based on the packet length. If the payload length is 8 or 40  
6065 bytes, the payload is in CREG format; otherwise it is in REG format.

6066 The description of data fields of this control packet is described in Table 163 and Figure 169. The meaning of  
6067 the packets differs depending on the direction of the packet. This packet interpretation is explained in Table  
6068 164. These packets are used during the registration and unregistration processes in a compatibility mode  
6069 network, as explained in Annex K.2.1 and K.2.2.

6070 The PKT.SID field is used in this control packet as the Switch where the Service Node is registering. The  
6071 PKT.LNID field is used in this control packet as the Local Node Identifier being assigned to the Service Node  
6072 during the registration process negotiation.

6073 The CREG.CAP\_PA field is used to indicate the packet aggregation capability as discussed in Section 4.3.7. In  
6074 the uplink direction, this field is an indication from the registering Terminal Node about its own capabilities.  
6075 For the Downlink response, the Base Node evaluates whether or not all the devices in the cascaded chain  
6076 from itself to this Terminal Node have packet-aggregation capability. If they do, the Base Node shall set  
6077 CREG.CAP\_PA=1; otherwise CREG.CAP\_PA=0.

MSB



LSB

Figure 169 - CREG control packet structure

Table 163 - CREG control packet fields

| Name   | Length | Description                                                                                                                                                 |
|--------|--------|-------------------------------------------------------------------------------------------------------------------------------------------------------------|
| CREG.N | 1 bit  | Negative <ul style="list-style-type: none"> <li>CREG.N=1 for the negative register;</li> <li>CREG.N=0 for the positive register.</li> </ul> (see Table 164) |

| Name        | Length | Description                                                                                                                                                                                                                                                                                                                                             |
|-------------|--------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| CREG.R      | 1 bit  | Roaming <ul style="list-style-type: none"> <li>• CREG.R=1 if Node already registered and wants to perform roaming to another Switch;</li> <li>• CREG.R=0 if Node not yet registered and wants to perform a clear registration process.</li> </ul>                                                                                                       |
| CREG.SPC    | 2 bits | Security Profile Capability for Data PDUs: <ul style="list-style-type: none"> <li>• CREG.SPC=0 No encryption capability;</li> <li>• CREG.SPC=1 Security profile 1 capable device;</li> <li>• CREG.SPC=2 Security profile 2 capable device);</li> <li>• CREG.SPC=3 Security profile 3 capable device (not yet specified).</li> </ul>                     |
| CREG.CAP_R  | 1 bit  | Robust Mode <p>CREG REQ</p> <p>1 if the node is using a robust link to join the network;</p> <p>0 if the node is using a non-robust link to join the network</p> <p>CREG RSP</p> <p>the value is set to the same as CREG REQ frame</p> <p>CREG ACK</p> <p>the value is set to the same as CREG REQ frame</p>                                            |
| CREG.CAP_14 | 1 bit  | PRIME v1.4 Backward Compatibility Mode Capable <p>1 (uplink) if the device is capable of using PRIME v1.4 backwards compatibility mode (i.e. this value is 1 for all PRIME v1.4 devices sending this message).</p> <p>1 (downlink) if the base node is acting in 1.4 backwards compatibility mode.</p> <p>0 if the device is a PRIME v1.3.6 device.</p> |
| CREG.CAP_SW | 1 bit  | Switch Capable <p>1 if the device is able to behave as a Switch Node;</p> <p>0 if the device is not.</p>                                                                                                                                                                                                                                                |

| Name             | Length | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 |
|------------------|--------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| CREG.CAP_PA      | 1 bit  | Packet Aggregation Capability<br><br>1 if the device has packet aggregation capability (uplink)<br>if the data transit path to the device has packet aggregation capability (Downlink)<br>0 otherwise.                                                                                                                                                                                                                                                                                                                                                                      |
| CREG.CAP_CFP     | 1 bit  | Contention Free Period Capability<br><br>1 if the device is able to perform the negotiation of the CFP;<br>0 if the device cannot use the Contention Free Period in a negotiated way.                                                                                                                                                                                                                                                                                                                                                                                       |
| CREG.CAP_DC      | 1 bit  | Direct Connection Capability<br><br>1 if the device is able to perform direct connections;<br>0 if the device is not able to perform direct connections.                                                                                                                                                                                                                                                                                                                                                                                                                    |
| CREG.CAP_MC      | 1 bit  | Multicast Capability<br><br>1 if the device is able to use multicast for its own communications;<br>0 if the device is not able to use multicast for its own communications.                                                                                                                                                                                                                                                                                                                                                                                                |
| CREG.CAP_PR<br>M | 1 bit  | PHY Robustness Management Capable<br><br>1 if the device is able to perform PHY Robustness Management;<br>0 if the device is not able to perform PHY Robustness Management.                                                                                                                                                                                                                                                                                                                                                                                                 |
| CREG.CAP_ARQ     | 1 bit  | ARQ Capable<br><br>1 if the device is able to establish ARQ connections;<br>0 if the device is not able to establish ARQ connections.                                                                                                                                                                                                                                                                                                                                                                                                                                       |
| CREG.TIME        | 3 bits | Time to wait for an ALV_B messages before assuming the Service Node has been unregistered by the Base Node. For all messages except REG_RSP this field should be set to 0. For REG_RSP its value means:<br><br>CALV.TIME = 0 => 32 seconds;<br>CALV.TIME = 1 => 64 seconds;<br>CALV.TIME = 2 => 128 seconds ~ 2.1 minutes;<br>CALV.TIME = 3 => 256 seconds ~ 4.2 minutes;<br>CALV.TIME = 4 => 512 seconds ~ 8.5 minutes;<br>CALV.TIME = 5 => 1024 seconds ~ 17.1 minutes;<br>CALV.TIME = 6 => 2048 seconds ~ 34.1 minutes;<br>CALV.TIME = 7 => 4096 seconds ~ 68.3 minutes. |
| CREG.EUI-48      | 48 bit | EUI-48 of the Node<br><br>EUI-48 of the Node requesting the Registration.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |

| Name     | Length   | Description                                                                                       |
|----------|----------|---------------------------------------------------------------------------------------------------|
| CREG.SNK | 128 bits | Encrypted Subnetwork key that shall be used to derive the Subnetwork working key                  |
| CREG.AUK | 128 bits | Encrypted authentication key. This is a random sequence meant to act as authentication mechanism. |

6081

Table 164 - CREG control packet types

| Name      | HDR.DO | CPKT.LNID | CREG.N | CREG.R | Description                                                                                                                                                                                                       |
|-----------|--------|-----------|--------|--------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| REG_REQ   | 0      | 0x3FFF    | 0      | R      | Registration request <ul style="list-style-type: none"> <li>If R=0 any previous connection from this Node should be lost;</li> <li>If R=1 any previous connection from this Node should be maintained.</li> </ul> |
| REG_RSP   | 1      | < 0x3FFF  | 0      | R      | Registration response. This packet assigns the CPCK.LNID to the Service Node.                                                                                                                                     |
| REG_ACK   | 0      | < 0x3FFF  | 0      | R      | Registration acknowledged by the Service Node.                                                                                                                                                                    |
| REG_REJ   | 1      | 0x3FFF    | 1      | 0      | Registration rejected by the Base Node.                                                                                                                                                                           |
| REG_UNR_S | 0      | < 0x3FFF  | 1      | 0      | <ul style="list-style-type: none"> <li>After a REG_UNR_B: Unregistration acknowledge;</li> <li>Alone: Unregistration request initiated by the Node.</li> </ul>                                                    |
| REG_UNR_B | 1      | < 0x3FFF  | 1      | 0      | <ul style="list-style-type: none"> <li>After a REG_UNR_S: Unregistration acknowledge;</li> <li>Alone: Unregistration request initiated by the Base Node</li> </ul>                                                |

6082

6083 Fields CREG.SNK and CREG.AUK are of significance only for REG\_RSP and REG\_ACK messages with Security  
6084 Profile 1 (CREG.SCP=1). For all other message-exchange variants using the CREG control packet, these fields  
6085 shall not be present reducing the length of payload.

6086 In REG\_RSP message, the CREG.SNK and CREG.AUK shall always be inserted encrypted with WK0.

6087 In the REG\_ACK message, the CREG.SNK field shall be set to zero. The contents of the CREG.AUK field shall  
6088 be derived by decrypting the received REG\_RSP message with WK0 and re-encrypting the decrypted  
6089 CREG.AUK field with SWK derived from the decrypted CREG.SNK and random sequence previously received  
6090 in SEC control packets.

### K.1.1.1.2 Compatibility PRO control packet (CPRO, CPKT.CTYPE = 3)

The compatibility promotion (CPRO) control packet is used by the base node and all service nodes to promote a Service Node from Terminal function to Switch function. The description of the fields of this packet is given in Table 165 and Figure 170. The meaning of the packet differs depending on the direction of the packet and on the values of the different types. Table 166 shows the different interpretation of the packets. The promotion process in backward compatibility mode is explained in more detail in Annex K.2.3 and K.2.3.1.

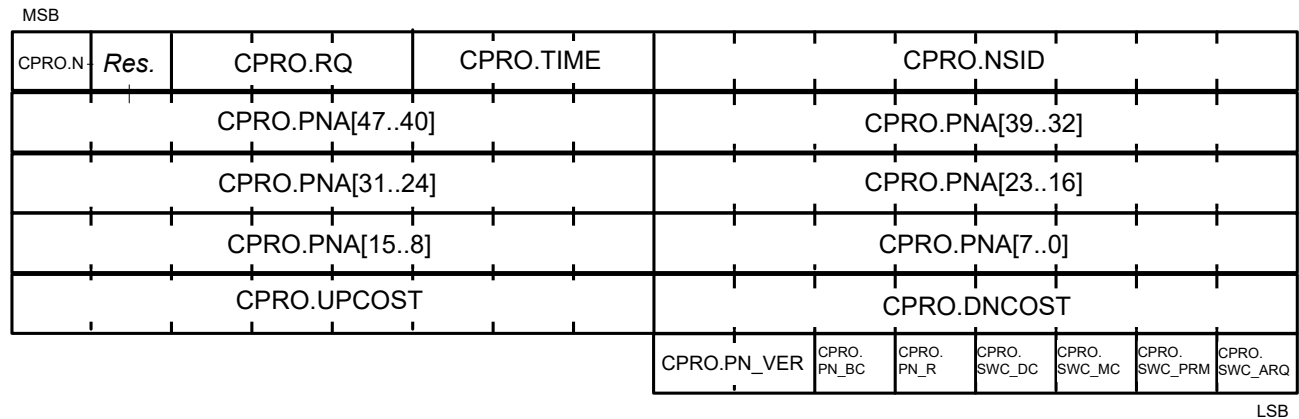


Figure 170 - CPRO\_REQ\_S control packet structure

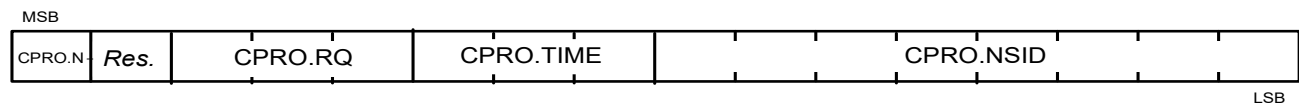


Figure 171 - CPRO control packet structure

Note that Figure 170 includes all fields as used by a CPRO\_REQ\_S message. All other messages are much smaller, containing only CPRO.N, CPRO.RC, CPRO.TIME and CPRO.NSID as shown in Figure 171.

Table 165 - CPRO control packet fields

| Name      | Length | Description                                                                                                                                                                                                       |
|-----------|--------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| CPRO.N    | 1 bit  | Negative<br>CPRO.N=1 for the negative promotion<br>CPRO.N=0 for the positive promotion                                                                                                                            |
| Reserved  | 1 bit  | Reserved for future version of this protocol<br>This shall be 0 for this version of the protocol.                                                                                                                 |
| CPRO.RQ   | 3 bits | Receive quality of the PNPDU message received from the Service Node requesting the Terminal to promote.                                                                                                           |
| CPRO.TIME | 3 bits | The ALV.TIME which is being used by the terminal which will become a switch. On a reception of this time in a PRO_REQ_B the Service Node should reset the Keep-Alive timer in the same way as receiving an ALV_B. |



| Name        | Length       | Description                                                                                                                                                                                                                                                                             |
|-------------|--------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| CPRO.NSID   | 8 bits       | <p>New Switch Identifier.</p> <p>This is the assigned Switch identifier of the Node whose promotion is being managed with this packet. This is not the same as the PKT.SID of the packet header, which must be the SID of the Switch this Node is connected to, as a Terminal Node.</p> |
| CPRO.PNA    | 0 or 48 bits | <p>Promotion Need Address contains the EUI-48 of the Terminal requesting the Service Node promotes to become a Switch.</p> <p>This field is only included in the PRO_REQ_S message.</p>                                                                                                 |
| CPRO.UPCOST | 0 or 8 bits  | <p>Total uplink cost from the Terminal Node to the Base Node. This value is calculated in the same way a Switch Node calculates the value it places into its own Beacon PDU.</p> <p>This field is only included in the PRO_REQ_S message.</p>                                           |
| CPRO.DNCOST | 0 or 8 bits  | <p>Total Downlink cost from the Base Node to the Terminal Node. This value is calculated in the same way a Switch Node calculates the value it places into its own Beacon PDU.</p> <p>This field is only included in the PRO_REQ_S message.</p>                                         |
| CPRO.PN_VER | 2 bits       | <p>Protocol version (PNH.VER) of the node represented by PRO.PNA.</p> <p>(This field is always zero for PRIME v1.3.6 nodes)</p>                                                                                                                                                         |
| CPRO.PN_BC  | 1 bit        | <p>Backwards Compatibility mode of the node represented by PRO.PNA.</p> <p>1 if the device is backwards compatible with 1.3.6 PRIME<br/>0 if it is not.</p> <p>(This field is always zero for PRIME v1.3.6 nodes)</p>                                                                   |
| CPRO.PN_R   | 1 bit        | <p>Robust mode compatibility of the node represented by PRO.PNA.</p> <p>1 if the device supports robust mode<br/>0 if it is not</p> <p>(This field is always zero for PRIME v1.3.6 nodes)</p>                                                                                           |
| CPRO.SWC_DC | 1 bit        | <p>Direct Connection Switching Capability</p> <p>1 if the device is able to behave as Direct Switch in direct connections.<br/>0 otherwise</p>                                                                                                                                          |

| Name             | Length | Description                                                                                                                                                                                                                        |
|------------------|--------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| CPRO.SWC_MC      | 1 bit  | <p>Multicast Switching Capability</p> <p>1 if the device is able to manage the multicast traffic when behaving as a Switch.</p> <p>0 otherwise</p>                                                                                 |
| CPRO.SWC_PR<br>M | 1 bit  | <p>PHY Robustness Management Switching Capability</p> <p>1 if the device is able to perform PRM for the Terminal Nodes when behaving as a Switch.</p> <p>0 if the device is not able to perform PRM when behaving as a Switch.</p> |
| CPRO.SWC_ARQ     | 1 bit  | <p>ARQ Buffering Switching Capability</p> <p>1 if the device is able to perform buffering for ARQ connections while switching.</p> <p>0 if the device is not able to perform buffering for ARQ connections while switching.</p>    |

6105

Table 166 - CPRO control packet types

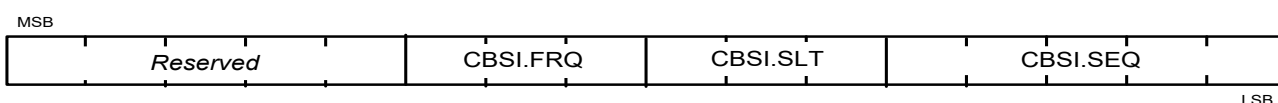
| Name      | HDR.DO | CPRO.N | CPRO.NSID | Description                                                                                                                                                                                                                                               |
|-----------|--------|--------|-----------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| PRO_REQ_S | 0      | 0      | 0xFF      | Promotion request initiated by the Service Node.                                                                                                                                                                                                          |
| PRO_REQ_B | 1      | 0      | < 0xFF    | <p>The Base Node will consider that the Service Node has promoted with the identifier CPRO.NSID.</p> <ul style="list-style-type: none"> <li>After a PRO_REQ: Promotion accepted;</li> <li>Alone: Promotion request initiated by the Base Node.</li> </ul> |
| PRO_ACK   | 0      | 0      | < 0xFF    | Promotion acknowledge                                                                                                                                                                                                                                     |
| PRO_REJ   | 1      | 1      | 0xFF      | The Base Node will consider that the Service Node is demoted. It is sent after a PRO_REQ to reject it.                                                                                                                                                    |
| PRO_DEM_S | 0      | 1      | < 0xFF    | <p>The Service Node considers that it is demoted:</p> <ul style="list-style-type: none"> <li>After a PRO_DEM_B: Demotion accepted;</li> <li>After a PRO_REQ_B: Promotion rejected;</li> <li>Alone: Demotion request.</li> </ul>                           |

| Name      | HDR.DO | CPRO.N | CPRO.NSID | Description                                                                                                                                                                                |
|-----------|--------|--------|-----------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| PRO_DEM_B | 1      | 1      | < 0xFF    | <p>The Base Node considers that the Service Node is demoted.</p> <ul style="list-style-type: none"> <li>After a PRO_DEM_S: Demotion accepted;</li> <li>Alone: Demotion request.</li> </ul> |

6106

### 6107 K.1.1.1.3 Compatibility BSI control packet (CBSI, CPKT.CTYPE = 4)

6108 The Compatibility Beacon Slot Information (CBSI) control packet is only used by the Base Node and Switch  
6109 Nodes. It is used to exchange information that is further used by a Switch Node to transmit its beacon. The  
6110 description of the fields of this packet is given in Table 167 and Figure 172. The meaning of the packet differs  
6111 depending on the direction of the packet and on the values of the different types. Table 168 represents the  
6112 different interpretation of the packets. The promotion process is explained in more detail in 4.6.3.



6113

6114

Figure 172 - CBSI control packet structure

6115

Table 167 - CBSI control packet fields

| Name     | Length | Description                                                                                                                                                                                                                                                                                                                                                                          |
|----------|--------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Reserved | 5 bits | Reserved for future version of this protocol. In this version, this field should be initialized to 0.                                                                                                                                                                                                                                                                                |
| CBSI.FRQ | 3 bits | <p>Transmission frequency of Beacon Slot, encoded as:</p> <p>FRQ = 0 =&gt; 1 beacon every frame<br/> FRQ = 1 =&gt; 1 beacon every 2 frames<br/> FRQ = 2 =&gt; 1 beacon every 4 frames<br/> FRQ = 3 =&gt; 1 beacon every 8 frames<br/> FRQ = 4 =&gt; 1 beacon every 16 frames<br/> FRQ = 5 =&gt; 1 beacon every 32 frames<br/> FRQ = 6 =&gt; Reserved<br/> FRQ = 7 =&gt; Reserved</p> |
| CBSI.SLT | 3 bits | <p>Beacon Slot to be used by target Switch</p> <p>0 – 4: non-robust mode beacon slot<br/> 5 – 6: robust mode beacon slot</p>                                                                                                                                                                                                                                                         |
| CBSI.SEQ | 5 bits | The Beacon Sequence number when the specified change takes effect.                                                                                                                                                                                                                                                                                                                   |

6116

Table 168 - CBSI control message types

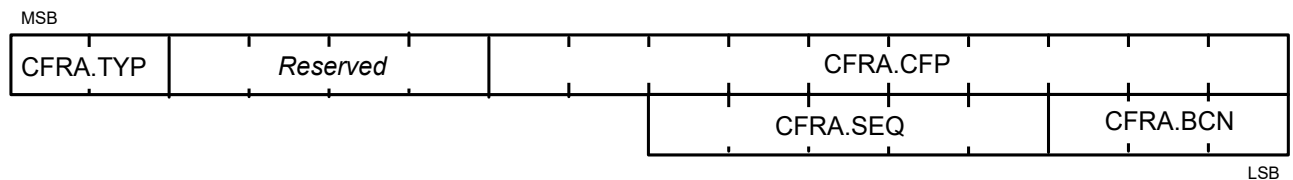
| Name    | HDR.DO | Description                                       |
|---------|--------|---------------------------------------------------|
| BSI_ACK | 0      | Acknowledgement of receipt of BSI control message |

| Name    | HDR.DO | Description                |
|---------|--------|----------------------------|
| BSI_IND | 1      | Beacon-slot change command |

6117

#### 6118 K.1.1.1.4 Compatibility FRA control packet (CFRA, CPKT.CTYPE = 5)

6119 This control packet is broadcast from the Base Node and relayed by all Switch Nodes to the entire  
 6120 Subnetwork. It is used by switches transmitting CBCN compatibility beacons, and the terminal nodes directly  
 6121 attached to them, to learn about upcoming frame changes. The description of fields of this packet is given in  
 6122 Table 169 and Figure 173. Table 170 shows the different interpretations of the packets.



6123

6124 Figure 173 - CFRA control packet structure

6125 Table 169 - CFRA control packet fields

| Name     | Length  | Description                                                                                                          |
|----------|---------|----------------------------------------------------------------------------------------------------------------------|
| CFRA.TYP | 2 bits  | 0: Beacon count change<br>1: CFP duration change<br>2: Reserved for PRIME v1.4 FRA message (see Section 4.4.2.6.5.1) |
| Reserved | 4 bits  | Reserved for future version of this protocol. In this version, this field should be initialized to 0.                |
| CFRA.CFP | 10 bits | Offset of CFP from start of frame                                                                                    |
| CFRA.SEQ | 5 bits  | The Beacon Sequence number when the specified change takes effect.                                                   |
| CFRA.BCN | 3 bits  | Number of beacons in a frame                                                                                         |

6126 Table 170 - CFRA control packet types

| Name        | CFRA.TYP | Description                                                                                                                                                            |
|-------------|----------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| FRA_BCN_IND | 0        | Indicates changes to frame structure due to change in beacon-slot count                                                                                                |
| FRA_CFP_IND | 1        | Indicates changes to frame structure due to change in CFP duration as a result of grant of CFP or end of CFP period for any requesting Service Node in the Subnetwork. |

#### 6127 K.1.1.1.5 Compatibility ALV control packet (CALV, CPKT.CTYPE = 7)

6128 In a compatibility mode network, the CALV control message is used exclusively for Keep-Alive signaling  
 6129 between a Service Node, the Service Nodes above it and the Base Node. The message exchange is

bidirectional, that is, a message is periodically exchanged in each direction. The structure of these messages is shown in Figure 174 and Table 171. The different Keep-Alive message types are shown in Table 172. The compatibility keep-alive process is shown in Annex K.2.5.

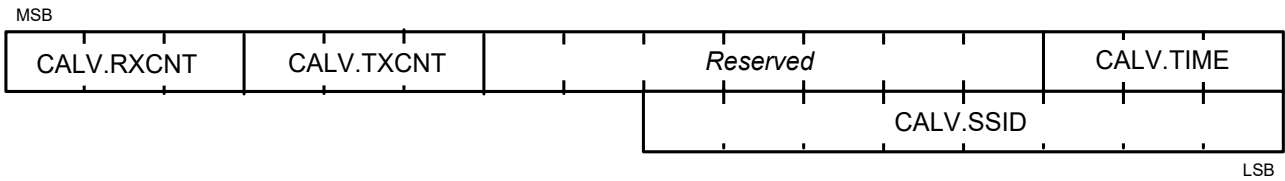


Figure 174 - CALV Control packet structure

Table 171 - CALV control message fields

| Name            | Length | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                 |
|-----------------|--------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| CALV.RXCNT      | 3 bits | Modulo 8 counter to indicate number of received CALV messages.                                                                                                                                                                                                                                                                                                                                                                                                              |
| CALV.TXCNT      | 3 bits | Modulo 8 counter to indicate number of transmitted CALV messages.                                                                                                                                                                                                                                                                                                                                                                                                           |
| <i>Reserved</i> | 7 bits | Should always be encoded as 0 in this version of the specification.                                                                                                                                                                                                                                                                                                                                                                                                         |
| CALV.TIME       | 3 bits | Time to wait for an ALV_B messages before assuming the Service Node has been unregistered by the Base Node.<br>CALV.TIME = 0 => 32 seconds;<br>CALV.TIME = 1 => 64 seconds;<br>CALV.TIME = 2 => 128 seconds ~ 2.1 minutes;<br>CALV.TIME = 3 => 256 seconds ~ 4.2 minutes;<br>CALV.TIME = 4 => 512 seconds ~ 8.5 minutes;<br>CALV.TIME = 5 => 1024 seconds ~ 17.1 minutes;<br>CALV.TIME = 6 => 2048 seconds ~ 34.1 minutes;<br>CALV.TIME = 7 => 4096 seconds ~ 68.3 minutes. |
| CALV.SSID       | 8 bits | For a Terminal, this should be 0xFF. For a Switch, this is its Switch Identifier.                                                                                                                                                                                                                                                                                                                                                                                           |

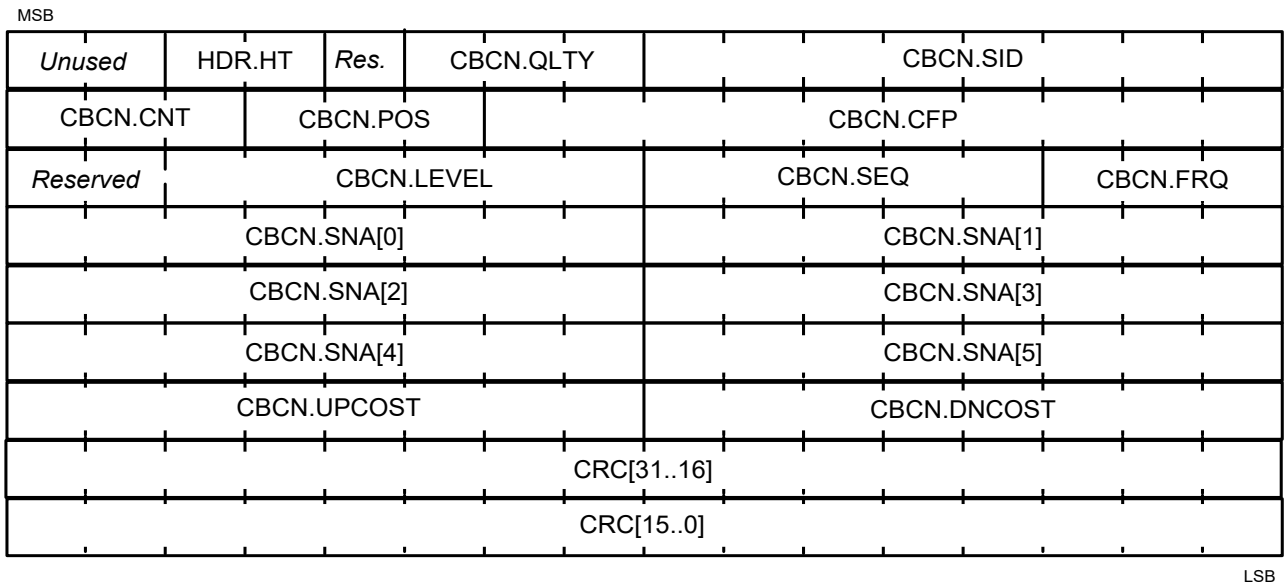
Table 172 – Keep-Alive control packet types

| Name  | HDR.DO | Description                            |
|-------|--------|----------------------------------------|
| ALV_S | 0      | Keep-Alive message from a Service Node |
| ALV_B | 1      | Keep-Alive message from the Base Node  |

#### K.1.2 Compatibility Beacon PDU (CBCN)

In a compatibility mode network, the compatibility beacon PDU (CBCN) is transmitted by the base node and some of the Switch devices on the Subnetwork (see table in section 4.9.3.2).

6144 Figure 175 below shows contents of a CBCN beacon.



6145

6146

Figure 175 – Beacon PDU structure

6147 Table 173 shows the CBCN PDU fields.

6148

Table 173 - Beacon PDU fields

| Name      | Length  | Description                                                                                                                                                                                                                   |
|-----------|---------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Unused    | 2 bits  | Unused bits which are always 0; included for alignment with MAC_H field in PPDU header (Fig 7, Section 3.3.3).                                                                                                                |
| HDR.HT    | 2 bits  | Header Type<br>HDR.HT = 2 for Beacon PDU                                                                                                                                                                                      |
| Reserved  | 1 bit   | Always 0 for this version of the specification. Reserved for future use.                                                                                                                                                      |
| CBCN.QLTY | 3 bits  | Quality of round-trip connectivity from this Switch Node to the Base Node. CBCN.QLTY=7 for best quality (Base Node or very good Switch Node), CBCN.QLTY=0 for worst quality (Switch having unstable connection to Subnetwork) |
| CBCN.SID  | 8 bits  | Switch identifier of transmitting Switch                                                                                                                                                                                      |
| CBCN.CNT  | 3 bits  | Number of beacon-slots in this frame                                                                                                                                                                                          |
| CBCN.SLT  | 3 bits  | Beacon-slot in which this BPDU is transmitted<br>CBCN.SLT=0 is reserved for the Base Node                                                                                                                                     |
| CBCN.CFP  | 10 bits | Offset of CFP from start of frame<br>CBCN.CFP=0 indicates absence of CFP in a frame.<br>(CBCN. CFP includes robust beacon slots)                                                                                              |

| Name        | Length  | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |
|-------------|---------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Reserved    | 1 bit   | Always 0 for this version of the specification. Reserved for future use.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |
| CBCN.LEVEL  | 6 bits  | Hierarchy of transmitting Switch in Subnetwork                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |
| CBCN.SEQ    | 5 bits  | Sequence number of this BPDU in super frame. Incremented for every beacon the Base Node sends and is propagated by Switch through its BPDU such that entire Subnetwork has the same notion of sequence number at a given time.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |
| CBCN.FRQ    | 3 bits  | <p>Transmission frequency of this BPDU. Values are interpreted as follows:</p> <ul style="list-style-type: none"> <li>0 = 1 beacon every frame</li> <li>1 = 1 beacon every 2 frames</li> <li>2 = 1 beacon every 4 frames</li> <li>3 = 1 beacon every 8 frames</li> <li>4 = 1 beacon every 16 frames</li> <li>5 = 1 beacon every 32 frames</li> <li>6 = Reserved</li> <li>7 = Reserved</li> </ul>                                                                                                                                                                                                                                                                                                                                                |
| CBCN.SNA    | 48 bits | Subnetwork identifier in which the Switch transmitting this BPDU is located                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |
| CBCN.UPCOST | 8 bits  | <p>Total uplink cost from the transmitting Switch Node to the Base Node. The cost of a single hop is calculated based on modulation scheme used on that hop in uplink direction. Values are derived as follows:</p> <ul style="list-style-type: none"> <li>8PSK = 0</li> <li>QPSK = 1</li> <li>BPSK = 2</li> <li>8PSK_F = 1</li> <li>QPSK_F = 2</li> <li>BPSK_F = 4</li> </ul> <p>The Base Node will transmit in its beacon a CBCN.UPCOST of 0. A Switch Node will transmit in its beacon the value of CBCN.UPCOST received from its upstream Switch Node, plus the cost of the upstream uplink hop to its upstream Switch. When this value is larger than what can be held in CBCN.UPCOST the maximum value of CBCN.UPCOST should be used.</p> |

| Name        | Length  | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             |
|-------------|---------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| CBCN.DNCOST | 8 bits  | <p>Total Downlink cost from the Base Node to the transmitting Switch Node. The cost of a single hop is calculated based on modulation scheme used on that hop in Downlink direction. Values are derived as follows:</p> <p>8PSK 0<br/>QPSK 1<br/>BPSK 2<br/>8PSK_F 1<br/>QPSK_F 2<br/>BPSK_F 4</p> <p>The Base Node will transmit in its beacon a CBCN.DNCOST of 0. A Switch Node will transmit in its beacon the value of CBCN.DNCOST received from its upstream Switch Node, plus the cost of the upstream Downlink hop from its upstream Switch. When this value is larger than what can be held in CBCN.DNCOST the maximum value of CBCN.DNCOST should be used.</p> |
| CRC         | 32 bits | The CRC shall be calculated with the same algorithm as the one defined for the CRC field of the MAC PDU (see section 4.4.2.4 for details). This CRC shall be calculated over the complete BPDU except for the CRC field itself. Since the SNA is present in the payload there is no concatenation of SNA and GPDU in this case.                                                                                                                                                                                                                                                                                                                                         |

6149

6150 The CBCN BPDU is also used to detect when the uplink Switch is no longer available either by a change in the  
6151 characteristics of the medium or because of failure etc. The rules in section 4.4.4 apply.

## 6152 K.2 MAC procedures

### 6153 K.2.1 Registration process

6154 The initial Service Node start-up (4.3.1) is followed by a Registration process. A Service Node in a  
6155 *Disconnected* functional state shall transmit a registration control packet to the Base Node in order to get  
6156 itself included in the Subnetwork.

6157 All beacons shall be sent in CBCN format.

6158 Since no LNID or SID is allocated to a Service Node at this stage, the CPKT.LNID field shall be set to all 1s and  
6159 the CPKT.SID field shall contain the SID of the Switch Node through which it seeks attachment to the  
6160 Subnetwork.

6161 Base Nodes may use a Registration request as an authentication mechanism. However this specification does  
6162 not recommend or forbid any specific authentication mechanism and leaves this choice to implementations.

6163 For all successfully accepted Registration requests, the Base Node shall allocate an LNID that is unique within  
6164 the domain of the Switch Node through which the attachment is realized. This LNID shall be indicated in the  
6165 PKT.LNID field of response (REG\_RSP). The assigned LNID, in combination with the SID of the Switch Node  
6166 through which the Service Node is registered, would form the NID of the registering Node.



Based on the flag CBNC.CAP\_14 a service node knows whether it is registered to a PRIME v1.4 (standard or compatibility mode) or a PRIME v1.3.6 network. Registration is a three-way process. The REG\_RSP shall be acknowledged by the receiving Service Node with a REG\_ACK message. The same format is used for the REG\_RSP as for the REG\_REQ.

Figure 176 represents a successful Registration process and Figure 177 shows a Registration request that is rejected by the Base Node. Details on specific fields that distinguish one Registration message from the other are given in Table 23 and Table 164.

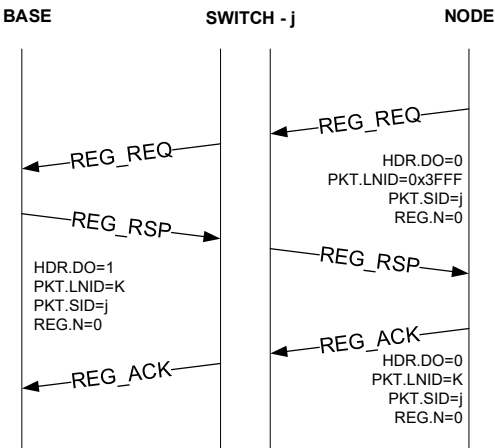


Figure 176 – Registration process accepted

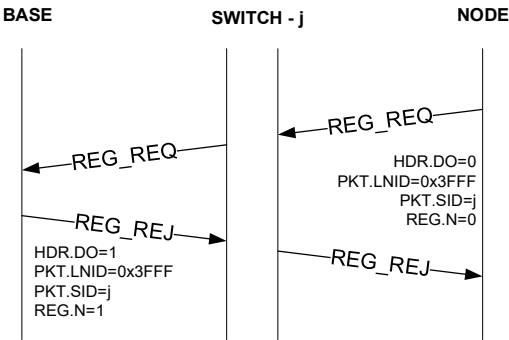


Figure 177 – Registration process rejected

When assigning an LNID, the Base Node shall not reuse an LNID released by an unregister process until after ( $macCtrlMsgFailTime + macMinCtlReTxTimer$ ) seconds, to ensure that all retransmit packets have left the Subnetwork. Similarly, the Base Node shall not reuse an LNID freed by the Keep-Alive process until  $T_{keep\_alive}$  seconds have passed, using the last known acknowledged  $T_{keep\_alive}$  value, or if larger, the last unacknowledged  $T_{keep\_alive}$ , for the Service Node using the LNID.

During network startup where the whole network is powered on at once, there will be considerable contention for the medium. It is recommended, but optional, that randomness is added to the first transmission of REQ\_REQ and all subsequent retransmissions. A random delay of maximum duration of 10% of  $macMinCtlReTxTimer$  may be imposed before the first REG\_REQ message, and a similar random delay of up to 10% of  $macMinCtlReTxTimer$  may be added to each retransmission.

**6189 K.2.2 Unregistering process**

6190 The unregistering process follows the description in Section 4.6.2. All nodes use compatibility mode  
6191 unregistration packets (CREG).

**6192 K.2.3 Promotion process**

6193 A Node that cannot reach any existing Switch may send promotion-needed frames so that a Terminal can be  
6194 promoted and begin to switch. During this process, a Node that cannot reach any existing Switch may send  
6195 PNPDUs so that a nearby Terminal can be promoted and begin to act as a Switch. During this process, a  
6196 Terminal will receive PNPDUs and at its discretion, generate compatibility mode PRO\_REQ control packets to  
6197 the Base Node. In a compatibility mode network no standard PRO messages are used but only CPRO and CBSI  
6198 messages.

6199 The Base Node examines the promotion requests during a period of time. It may use the address of the new  
6200 Terminal, provided in the promotion-request packet, to decide whether or not to accept the promotion. It  
6201 will decide which Node shall be promoted, if any, sending a promotion response. The other Nodes will not  
6202 receive any answer to the promotion request to avoid Subnetwork saturation. Eventually, the Base Node  
6203 may send a rejection if any special situation occurs. If the Subnetwork is specially preconfigured, the Base  
6204 Node may send Terminal Node promotion requests directly to a Terminal Node.

6205 When a Terminal Node requests promotion, the CPRO.NSID field in the PRO\_REQ\_S message shall be set to  
6206 all 1s. The PRO.NSID field shall contain an LSID allocated to the promoted Node in the PRO\_REQ\_B message.  
6207 The acknowledging Switch Node shall set the CPRO.NSID field in its PRO\_ACK to the newly allocated LSID.  
6208 This final PRO\_ACK shall be used by intermediate Switch Nodes to update their switching tables as described  
6209 in 4.3.5.2.

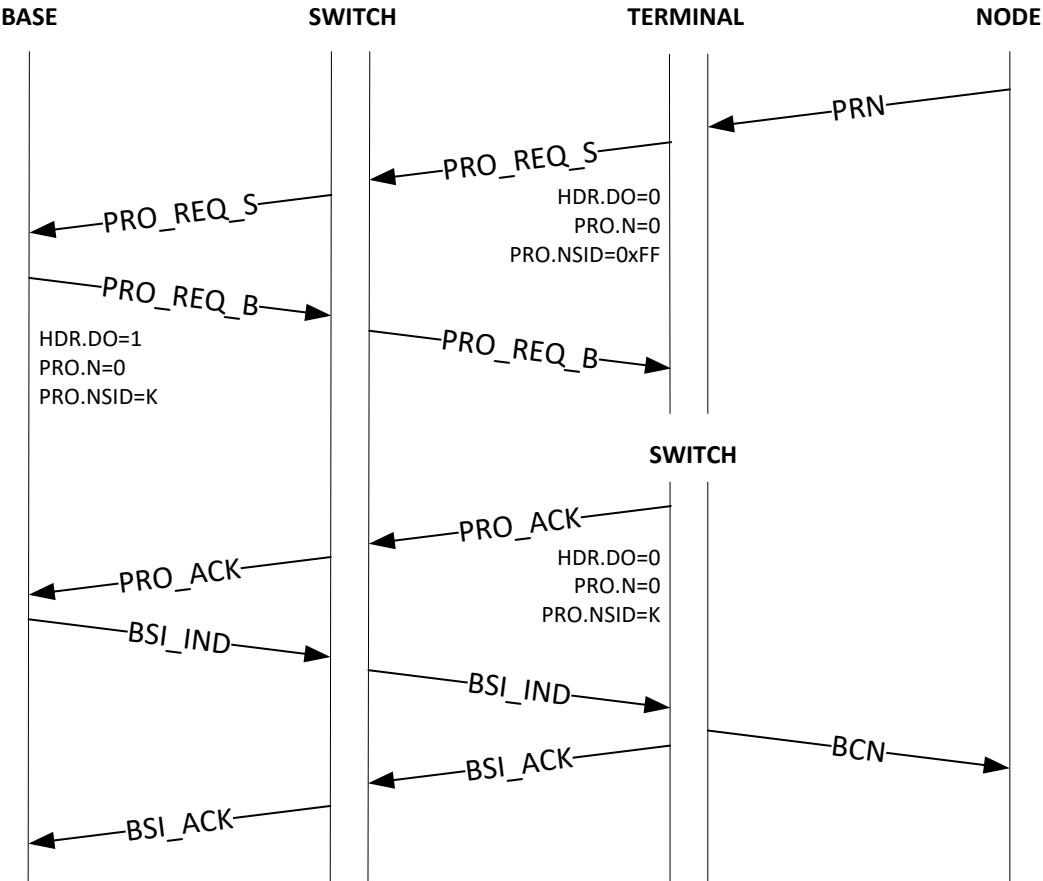
6210 When reusing LSIDs that have been released by a demotion process, the Base Node should not allocate the  
6211 LSID until after  $(macCtrlMsgFailTime + macMinCtrlReTxTimer)$  seconds to ensure all retransmit packets that  
6212 might use that LSID have left the Subnetwork. Similarly, the Base Node shall not reuse an LNID freed by the  
6213 Keep-Alive process until  $T_{keep\_alive}$  seconds have passed, using the last known acknowledged  $T_{keep\_alive}$  value,  
6214 or if larger, the last unacknowledged  $T_{keep\_alive}$ , for the Service Node using the LNID.

6215 After the base node receives the PRO\_ACK, the Base Node sends a BSI\_IND to the service node. The encoding  
6216 of the Beacon is decided using the beacon slot, if the beacon slot is 5 or 6, the encoding shall be DBPSK\_R.  
6217 The service node shall respond with the corresponding BSI\_ACK.

6218 The base node can use BSI\_IND with two purposes:

- 6219 • Change the allocation of the transmitted beacon. Only if the robustness of the beacon does not  
6220 change.
- 6221 • Start double switching by sending a second beacon in the other modulation.
- 6222 • After a switch is double switching the next BSI\_IND shall change the transmission properties of the  
6223 robust beacon if slot is 5 or 6 and of the non-robust beacon otherwise.

6224

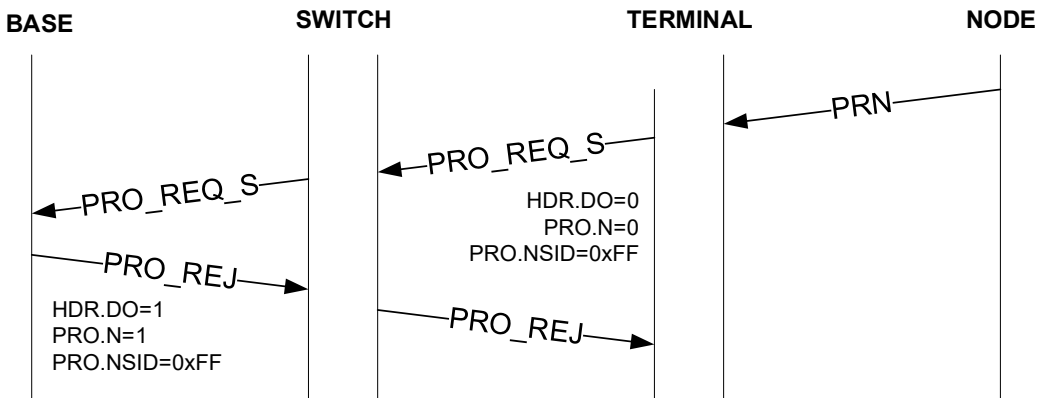


6225

6226

Figure 178 – Promotion process initiated by a Service Node

6227



6228

6229

Figure 179 – Promotion process rejected by the Base Node

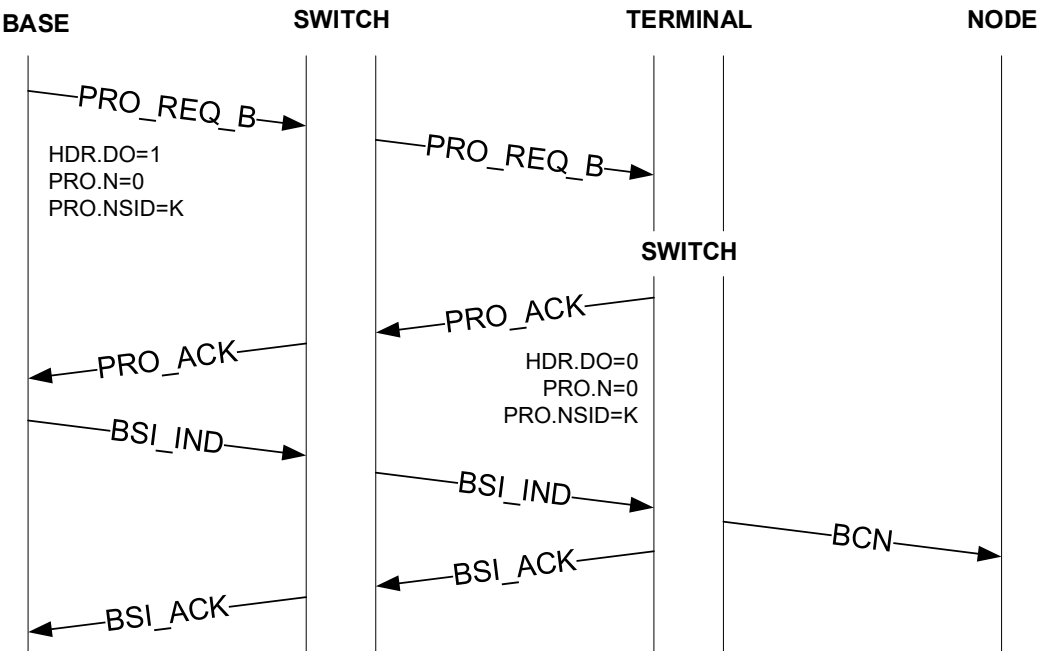


Figure 180 – Promotion process initiated by the Base Node

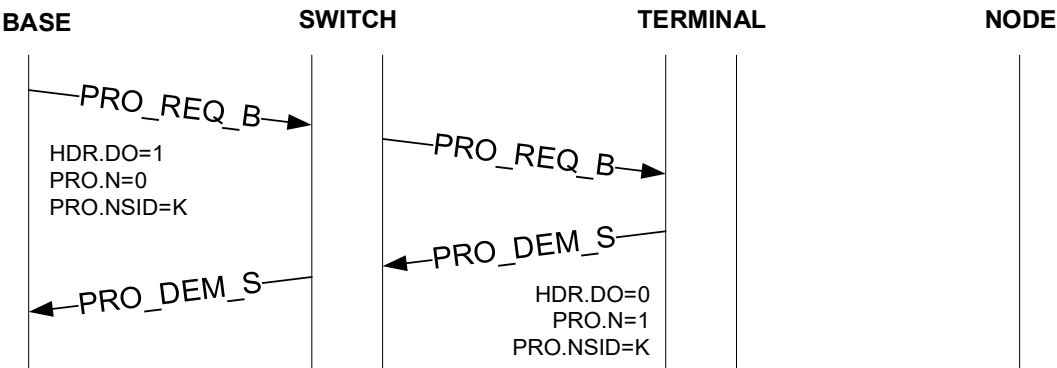


Figure 181 – Promotion process rejected by a Service Node

**K.2.3.1 Double switching**

Every time a Base Node promotes a node to act as robust switch, it shall start two BSI procedures to promote the Service Node so it has two beacon slots assigned, one robust and one non-robust.

One of the BSI\_IND shall have a beacon slot in the range 0-4 for the non-robust beacon (DBPSK\_CC) and the other one shall send the beacon slot in the range 5-6 for the robust beacon (DBPSK\_R). For future changes of the BSI information the rule to separate the robust and non-robust beacons shall be the range of the beacon slot

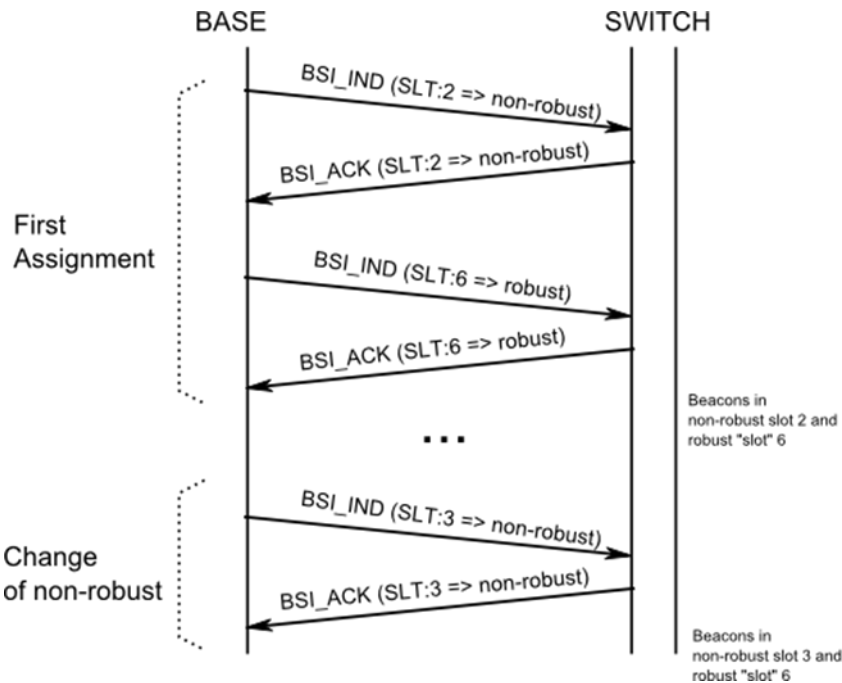


Figure 182 - Double switching BSI message exchange

**K.2.4 Demotion process**

The Base Node or a Switch Node may decide to discontinue a switching function at anytime. The demotion process provides for such a mechanism. In a compatibility mode network, only CPRO control packets are used for all demotion transactions.

The CPRO.NSID field shall contain the SID of the Switch Node that is being demoted as part of the demotion transaction. The PRO.PNA field is not used in any demotion process transaction and its contents are not interpreted at either end.

Following the successful completion of a demotion process, a Switch Node shall immediately stop the transmission of beacons and change from a *Switch* functional state to a *Terminal* functional state. The Base Node may reallocate the LSID and Beacon Slot used by the demoted Switch after (*macCtrlMsgFailTime* + *macMinCtlReTxTimer*) seconds to other Terminal Nodes requesting promotion.

The present version of this specification does not specify any explicit message to reject a demotion requested by a peer at the other end.

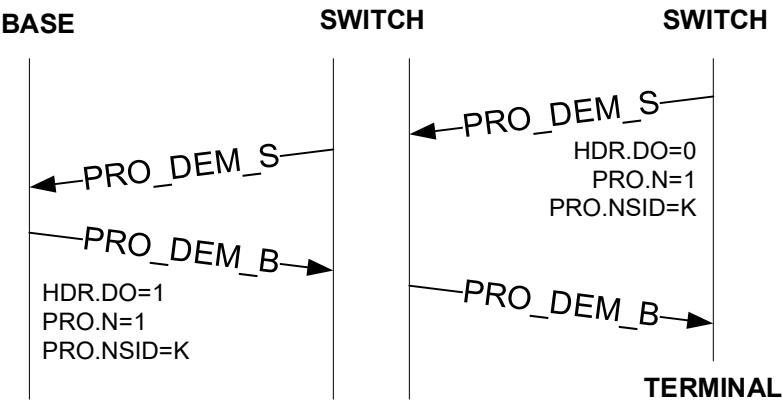


Figure 183 – Demotion process initiated by a Service Node

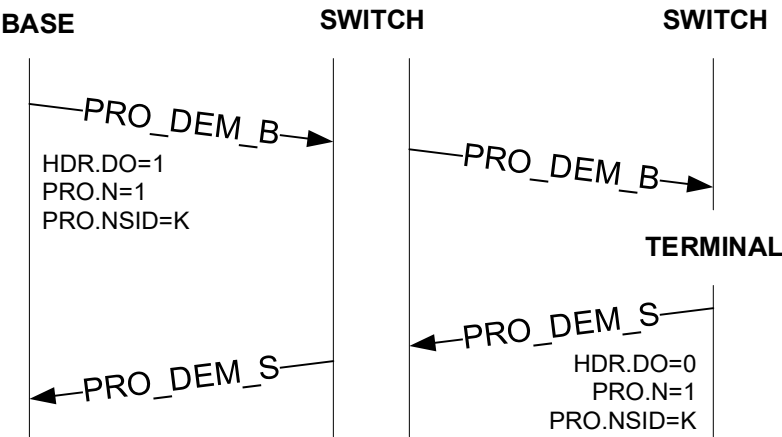


Figure 184 – Demotion process initiated by the Base Node

**K.2.5 Keep-Alive process**

The Keep-Alive process in a compatibility mode network is fundamentally different from the Keep-Alive process used in a standard PRIME v1.4 network. It is based on the PRIME v1.3.6 end-to-end process. The Keep-Alive process is used to detect when a Service Node has left the Subnetwork because of changes to the network configuration or because of fatal errors it cannot recover from.

When the Service Node receives the REG\_RSP packet it uses the REG.TIME/CREG.TIME field to start a timer  $T_{keep\_alive}$ . For every ALV\_B it receives, it restarts this timer using the value from CALV.TIME. The encoding of CALV.TIME is specified in Table 171. It should also send an ALV\_S to the Base Node. If the timer ever expires, the Service Node assumes it has been unregistered by the Base Node. The message PRO\_REQ does also reset the Keep-Alive timer to the CPRO.TIME value.

Each switch along the path of a ALV\_B message takes should keep a copy of the CPRO.TIME and then CALV.TIME for each Switch Node below it in the tree. When the switch does not receive an ALV\_S message from a Service Node below it for  $T_{keep\_alive}$  as defined in CPRO.TIME and CALV.TIME it should remove the Switch Node entry from its switch table. See section 4.3.5.2 for more information on the switching table. Additionally a Switch Node may use the REG.TIME/CREG.TIME and CALV.TIME to consider also every Service Node Registration status and take it into account for the switching table.

For every ALV\_S or ALV\_B message sent by the Base Node or Service Node, the counter CALV.TXCNT should be incremented before the message is sent. This counter is expected to wrap around. For every ALV\_B or

6278 ALV\_S message received by the Service Node or the Base Node the counter CALV.RXCNT should be  
6279 incremented. This counter is also expected to wrap around. These two counters are placed into the ALV\_S  
6280 and ALV\_B messages. The Base Node should keep a CALV.TXCNT and CALV.RXCNT separated counter for each  
6281 Service Node. These counters are reset to zero in the Registration process.

6282 The algorithm used by the Base Node to determine when to send ALV\_B messages to registered Service  
6283 Nodes and how to determine the value CALV.TIME and REG.TIME/CREG.TIME is not specified here.

6284 **K.2.6 Connection management**

6285 The processes follow the standard processes described in section 4.3

6286 **K.2.7 Multicast group management**

6287 The processes follow the standard processes described in section 4.6.7. The base node shall not send any  
6288 MUL\_SW\_LEAVE\_B to PRIME v1.3.6 service nodes, as the PRIME v1.3.6 switches implement a different  
6289 mechanism for multicast group tracking.

6290 **K.2.8 Robustness Management**

6291 Robustness management is not performed between devices running legacy version of protocol.

6292 **K.2.9 Channel allocation and deallocation**

6293 The process follows the description in Section 4.6.9. The Base Node shall send a CFRA broadcast packet.

## Annex L (informative)

### Type A, Type B PHY frames and Robust modes

The following is a recommendation about how to combine the two PHY frame formats defined by PRIME with the available payload transmission schemes. As a general guideline, preamble and header shall be at least as robust as the payload.

Type A and Type B PRIME PHY frames specify different Preamble lengths and Header formats:

- TYPE A PHY frames, as described in Figure 3, comprise a "Preamble A" lasting 2.048 ms and a "Header A" with a length equal to two OFDM symbols (2 x 2.24 ms).
- TYPE B PHY frames, as described in Figure 4, **achieve higher robustness** by means of a "Preamble B" lasting 8.192 ms and a "Header B" with a length equal to four OFDM symbols (4 x 2.24 ms)

Table 174 shows all possible combinations, recommendations [OK / NOK] are based on the fact that preamble and header shall be at least as robust as the payload.

Table 174 - PHY frame types and Payload transmission schemes

| HEADER<br>and<br>PREAMBLE                   | PAYLOAD      |              |          |       |          |       |          |       |
|---------------------------------------------|--------------|--------------|----------|-------|----------|-------|----------|-------|
|                                             | Robust DBPSK | Robust DQPSK | DBPSK_CC | DBPSK | DQPSK_CC | DQPSK | D8PSK_CC | D8PSK |
| Type A<br>(short preamble,<br>short header) | NOK          | NOK          | OK       | OK    | OK       | OK    | OK       | OK    |
| Type B<br>(long preamble,<br>long header)   | OK           | OK           | OK       | OK    | OK       | OK    | OK       | OK    |



Annex M  
(informative)  
Channel Hopping examples

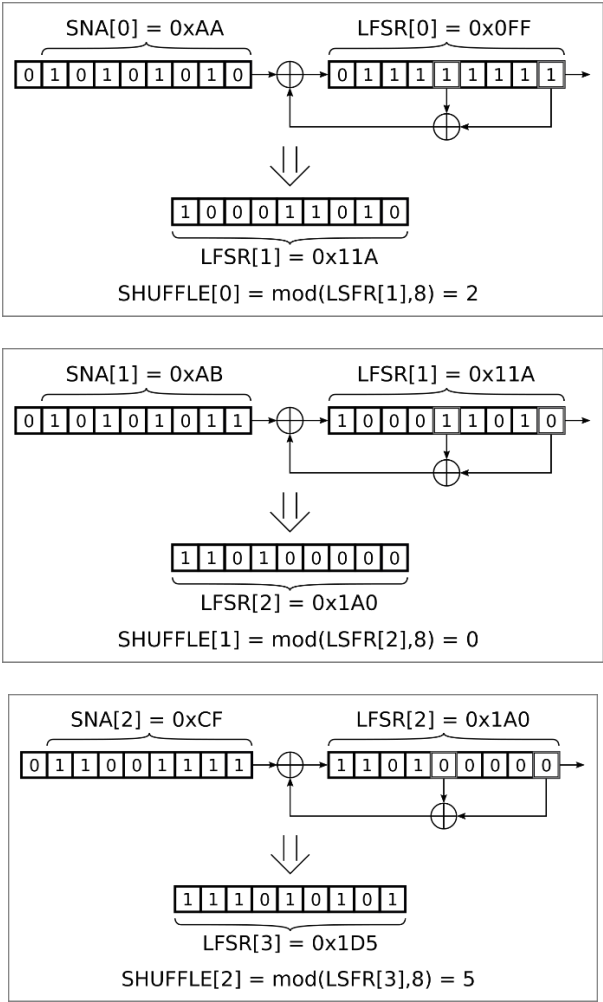
M.1 Channel sequence generation examples

In this section examples are reported for the generation of the main hopping sequence. For the sake of simplicity and to make easier the understanding, the examples use a contained number of channels. Moreover, note that the examples are also applicable to understand the generation of the beacon hopping sequence (with the due changes, e.g. macHoppingSequenceLength => macHoppingBCNSequenceLength, macHoppingInitialChannelList => macHoppingBCNInitialChannelList).

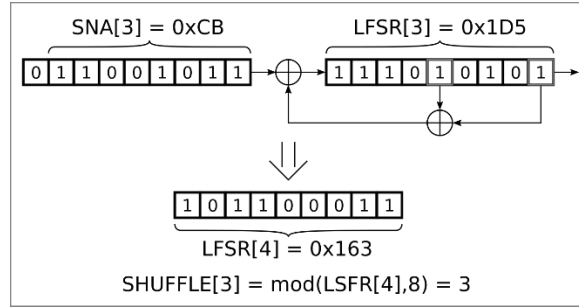
M.1.1 First example

For this example we will be generating the channel sequence of a RF medium with macHoppingInitialChannelList = 0x00FF, hence macHoppingSequenceLength equal to 8, and a SNA with value 7A:2B:CB:CF:AB:AA.

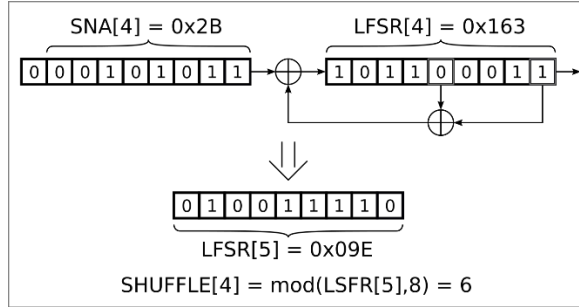
The first step is to obtain the SHUFFLE array values:



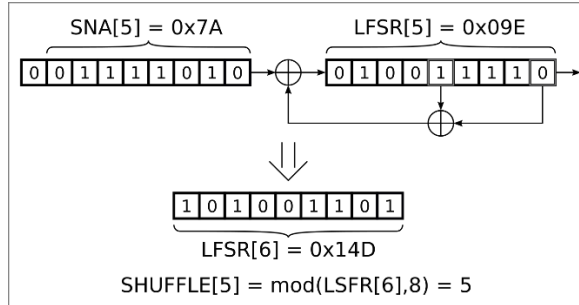
6327



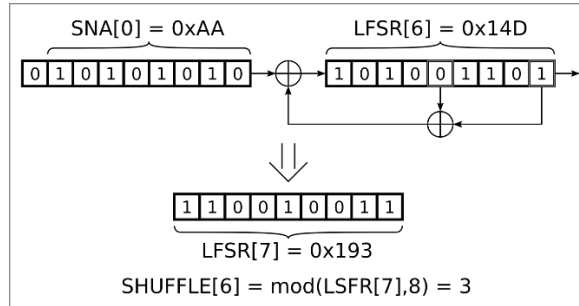
6328



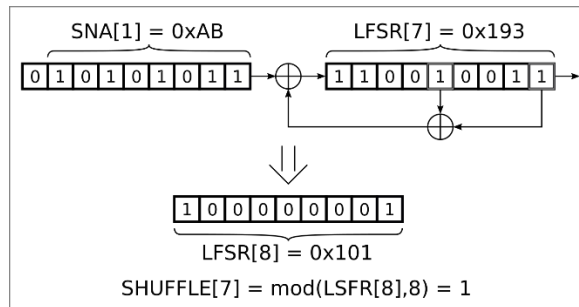
6329



6330



6331



6332 Then apply the shuffling algorithm to the CHANNELS array:

6333

SHUFFLE[] = { 2, 0, 5, 3, 6, 5, 3, 1 }  
CHANNELS[] = { 0, 1, 2, 3, 4, 5, 6, 7 }

6334

SHUFFLE[] = { 2, 0, 5, 3, 6, 5, 3, 1 }  
CHANNELS[] = { 0, 1, 2, 3, 4, 5, 6, 7 }



6335

SHUFFLE[] = { 2, 0, 5, 3, 6, 5, 3, 1 }  
CHANNELS[] = { 2, 1, 0, 3, 4, 5, 6, 7 }



6336

SHUFFLE[] = { 2, 0, 5, 3, 6, 5, 3, 1 }  
CHANNELS[] = { 1, 2, 0, 3, 4, 5, 6, 7 }



6337

SHUFFLE[] = { 2, 0, 5, 3, 6, 5, 3, 1 }  
CHANNELS[] = { 1, 2, 5, 3, 4, 0, 6, 7 }



6338

SHUFFLE[] = { 2, 0, 5, 3, 6, 5, 3, 1 }  
CHANNELS[] = { 1, 2, 5, 3, 4, 0, 6, 7 }



6339

SHUFFLE[] = { 2, 0, 5, 3, 6, 5, 3, 1 }  
CHANNELS[] = { 1, 2, 5, 3, 6, 0, 4, 7 }



6340

SHUFFLE[] = { 2, 0, 5, 3, 6, 5, 3, 1 }  
CHANNELS[] = { 1, 2, 5, 3, 6, 0, 4, 7 }



6341

SHUFFLE[] = { 2, 0, 5, 3, 6, 5, 3, 1 }  
CHANNELS[] = { 1, 2, 5, 4, 6, 0, 3, 7 }



6342

SHUFFLE[] = { 2, 0, 5, 3, 6, 5, 3, 1 }  
CHANNELS[] = { 1, 7, 5, 4, 6, 0, 3, 2 }

6343 This resulting CHANNELS array will be the channel sequence shared by the network.

#### 6344 M.1.2 Second example

6345 In this section the example of M.1.1 is largely reused.

6346 Now, let's consider the case where macHoppingInitialChannelList = 0x03ED, hence again  
6347 macHoppingSequenceLength = 8, and a SNA with value 7A:2B:CB:CF:AB:AA. Compared to example of M.1.1,  
6348 macHoppingInitialChannelList is different, i.e. different channels are selected for hopping. Repeating the  
6349 same procedure reported in M.1.1, and summarizing the key steps, before shuffling we have

6350 SHUFFLE = {2, 0, 5, 3, 6, 5, 3, 1}

6351 CHANNELS = {0, 2, 3, 5, 6, 7, 8, 9}

6352 and after shuffling

6353 SHUFFLE = {2, 0, 5, 3, 6, 5, 3, 1}

6354 CHANNELS = {2, 9, 7, 6, 8, 0, 5, 3}

6355 **M.2 Channel transitions with different frame and CFP/SCP lengths: examples**

6356 Two examples are shown: one with frame length equal to 276 symbols (see Figure 185) and one with frame  
6357 length equal to 552 symbols (see Figure 186). For clarity, on both examples, two consecutive frames are  
6358 reported.

6359 Each example is subdivided into three parts: on the top row, the case where the CFP is smaller than 138  
6360 symbols, on the middle row, the case where the CFP is exactly 138 symbols, on the bottom row the case  
6361 where the CFP is greater than 138 symbols.

6362 According to section 4.6.11, the dwell time for the main sequence is 138 symbols, while the dwell time for  
6363 the beacon sequence is equal to the frame length. Guard-times are reported in grey in the figures: they also  
6364 allows identifying a RF channel change.

6365 In the examples, for ease of drawing, for the main sequence “Main Pos” is an abbreviation to avoid writing  
6366 macHoppingSequencePosition (and neglecting the formula that includes modulo  
6367 macHoppingSequenceLength). Similarly, for the beacon sequence, “FrameSeq” is the value incremented for  
6368 the beacon sequence (and neglecting the formula with BCN.SEQ modulo macHoppingBCNSequenceLength).

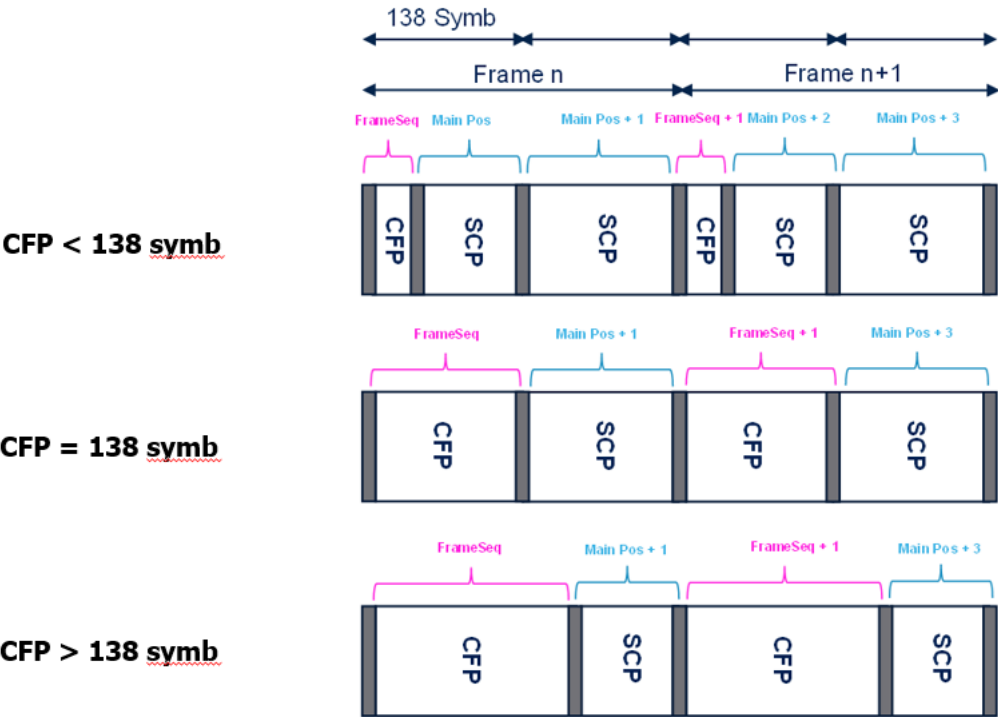


Figure 185 - Example with FrameLength = 276 symbols

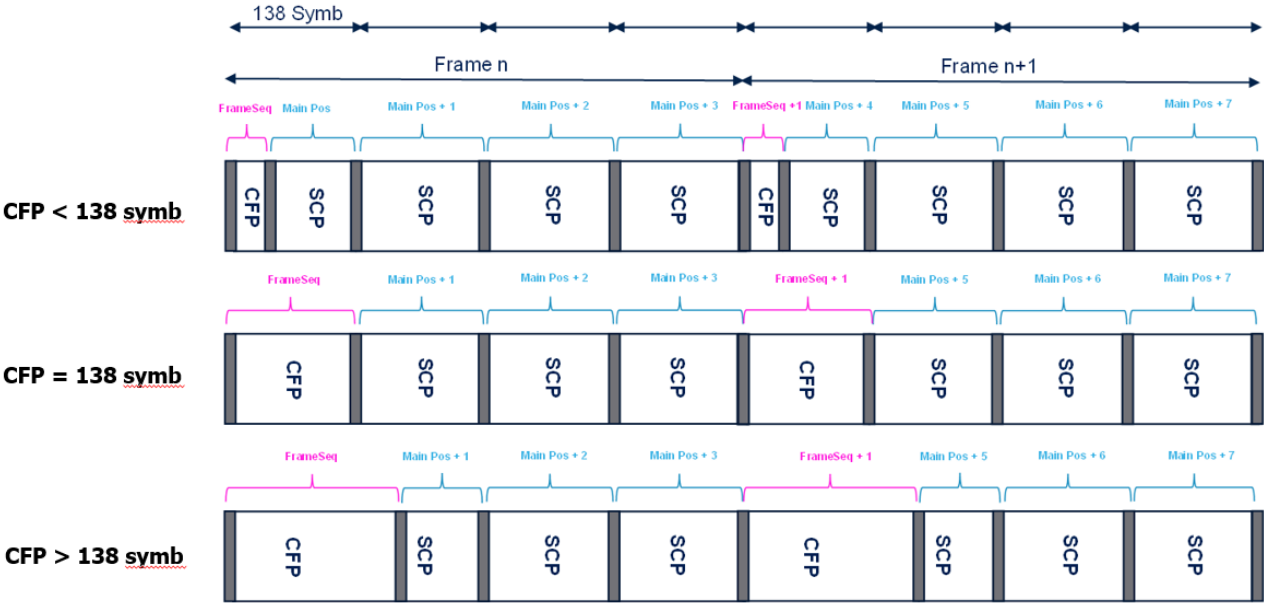


Figure 186 - Example with FrameLength = 552 symbols

As it can be seen, RF channel changes occur in these events:

- a transition between SCP and CFP (at the start of a new frame, at the first CFP symbol, swapping from the main channel sequence to the beacon channel sequence)
- a transition between CFP and SCP (at the last CFP symbol, swapping from the beacon channel sequence to the main channel sequence)
- after each dwell time during the SCP (channel changes in the main channel sequence)

**Annex N**  
**(normative)**  
**Management Information Base for PRIME Nodes**

The purpose of this annex is to describe the Management Information Base (MIB) required for every PRIME Base Node. The objects defined in this MIB will be requested by a Network Management System which will generate alarms and reports used for network operation purposes.

A part of MIB objects can be directly mapped to some existing PIB attributes. The value of the rest of MIB objects can be computed based on already defined PIB and List attributes.

**N.1 Basic requirements for Base Nodes**

**N.1.1 Instantaneous Objects**

The following objects should be available as instantaneous values to be consulted at any time:

| Object Name                                                            | Type      | Description                                                                                                                                                  |
|------------------------------------------------------------------------|-----------|--------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Network Uptime (ms)                                                    | Integer32 | Time from last network reset.<br><br>Network reset time is when the BN started to send beacons                                                               |
| Total number of Terminals in the subnetwork                            | Integer16 | Number of registered SNs in Terminal state                                                                                                                   |
| Total number of Switches in the subnetwork                             | Integer16 | Number of registered SNs in Switch state                                                                                                                     |
| Number of Service Nodes in the subnetwork connected through PLC medium | Integer16 | Number of registered SNs in Terminal or Switch state connected through PLC medium. Optional if Base Node is RF-only and Multi-PHY promotion is not supported |
| Number of Service Nodes in the subnetwork connected through RF medium  | Integer16 | Number of registered SNs in Terminal or Switch state connected through RF medium. Optional if Base Node is PLC-only and Multi-PHY promotion is not supported |
| Detail of Promoted Switches in the subnetwork                          | Table     | Number of Switches that are promoted in the subnetwork per each existing medium type. Optional table if Base Node is PLC-                                    |

|                                           |           |                                                                                                                                                                                                                                                                                                                                                    |           |
|-------------------------------------------|-----------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-----------|
|                                           |           | only or RF-only, and the Multi-PHY promotion is not supported.                                                                                                                                                                                                                                                                                     |           |
|                                           |           | Entry Element                                                                                                                                                                                                                                                                                                                                      | Type      |
|                                           |           | Number of Switches promoted only in PLC medium                                                                                                                                                                                                                                                                                                     | Integer16 |
|                                           |           | Number of Switches promoted only in RF medium                                                                                                                                                                                                                                                                                                      | Integer16 |
|                                           |           | Number of Switches promoted in both media (PLC+RF)                                                                                                                                                                                                                                                                                                 | Integer16 |
| Number of Beacons Allocated in PLC medium | Integer16 | Allocated PLC beacons in a superframe, calculated as the sum of all PLC beacons transmitted in the Subnetwork during a superframe. The number of PLC beacons transmitted by a device in a superframe depends on the transmission frequency (BCN.FRQ) associated to BPDU. Optional if Base Node is RF-only and Multi-PHY promotion is not supported |           |
| Number of Beacons Allocated in RF medium  | Integer16 | Allocated RF beacons in a superframe, calculated as the sum of all RF beacons transmitted in the Subnetwork during a superframe. The number of RF beacons transmitted by a device in a superframe depends on the transmission frequency (BCN.FRQ) associated to BPDU. Optional if Base Node is PLC-only and Multi-PHY promotion is not supported   |           |
| Number of Switching levels                | Integer16 | Number of logical levels in the subnetwork                                                                                                                                                                                                                                                                                                         |           |
| Number of Elements per level              | Table     | This object is a table with up to 63 rows (only 8 rows are required) and two columns.                                                                                                                                                                                                                                                              |           |

|                                                        |           |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |               |      |                         |           |                              |           |
|--------------------------------------------------------|-----------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|---------------|------|-------------------------|-----------|------------------------------|-----------|
|                                                        |           | <table><tr><td>Entry Element</td><td>Type</td></tr><tr><td>Number of Switches</td><td>Integer16</td></tr><tr><td>Number of Terminals</td><td>Integer16</td></tr></table>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         | Entry Element | Type | Number of Switches      | Integer16 | Number of Terminals          | Integer16 |
| Entry Element                                          | Type      |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |               |      |                         |           |                              |           |
| Number of Switches                                     | Integer16 |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |               |      |                         |           |                              |           |
| Number of Terminals                                    | Integer16 |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |               |      |                         |           |                              |           |
| Total Number of Unicast Active Connections             | Integer16 | The number of unicast non direct connections                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |               |      |                         |           |                              |           |
| Detail of Active Connections                           | Table     | <p>Number of active connections per each type of connection implemented in the Base Node and defined in Annex E of PRIME specification. If the BN does not implement a type of connection, it will answer with the message “No Such Object Available” for this object. If there are no active connections of a particular type, the associated Number of Active Connections will be zero. The minimum required objects are the ones associated to TYPE_CL_432 and TYPE_CL_MGMT</p> <table><tr><td>Entry Element</td><td>Type</td></tr><tr><td>Connection Type</td><td>Integer8</td></tr><tr><td>Number of Active Connections</td><td>Integer16</td></tr></table> | Entry Element | Type | Connection Type         | Integer8  | Number of Active Connections | Integer16 |
| Entry Element                                          | Type      |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |               |      |                         |           |                              |           |
| Connection Type                                        | Integer8  |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |               |      |                         |           |                              |           |
| Number of Active Connections                           | Integer16 |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |               |      |                         |           |                              |           |
| Total Number of Multicast Connections                  | Integer16 | Number of active multicast connections                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           |               |      |                         |           |                              |           |
| Detail of Devices Registered in a Multicast Connection | Table     | <p>Number of nodes per active multicast connection</p> <table><tr><td>Entry Element</td><td>Type</td></tr><tr><td>Multicast Connection ID</td><td>Integer16</td></tr><tr><td>Number of Devices per</td><td>Integer16</td></tr></table>                                                                                                                                                                                                                                                                                                                                                                                                                           | Entry Element | Type | Multicast Connection ID | Integer16 | Number of Devices per        | Integer16 |
| Entry Element                                          | Type      |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |               |      |                         |           |                              |           |
| Multicast Connection ID                                | Integer16 |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |               |      |                         |           |                              |           |
| Number of Devices per                                  | Integer16 |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |               |      |                         |           |                              |           |



|  |  |                      |  |
|--|--|----------------------|--|
|  |  | Multicast connection |  |
|--|--|----------------------|--|

**N.1.2 Periodic Objects**

The periodic objects' values should be integrated per a period equal to 1 minute. The Base Node should be able to keep the information for 60 periods. Apart from these periods, the Base Node should save the accumulated value for each object since the last network reset. The first row is the accumulated value, the second is the last period, the third is the period before that and so on.

The following objects should be available with the conditions described in the previous paragraph. Therefore, it is a table with 61 rows and 7 or 11 columns.

| Object Name                                                              | Type         | Description                                                                                                                                                                                                                                                                      |
|--------------------------------------------------------------------------|--------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Number of Topology Changes in Service Nodes connected through PLC medium | 4x Integer32 | <ul style="list-style-type: none"> <li>Promotions</li> <li>Demotions</li> <li>Registrations</li> <li>Unregistrations</li> </ul> <p>Optional if Base Node is RF-only and Multi-PHY promotion is not supported</p>                                                                 |
| Number of Topology Changes in Service Nodes connected through RF medium  | 4x Integer32 | <ul style="list-style-type: none"> <li>Promotions</li> <li>Demotions</li> <li>Registrations</li> <li>Unregistrations</li> </ul> <p>Optional if Base Node is PLC-only and Multi-PHY promotion is not supported</p>                                                                |
| Overall Coverage                                                         | Integer16    | It reflects the overall network reachability.                                                                                                                                                                                                                                    |
| Average Availability                                                     | Integer32    | The accumulated value (first value of the table) is calculated as the average availability of the different nodes registered (at least once in the period) in the subnetwork. The accumulated value (first row) will take into account every SN registered since the last reset. |

|                                                      |           |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |
|------------------------------------------------------|-----------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|                                                      |           | <p>The formulas to calculate these values are the following.</p> <p>For the accumulated value:</p> $Availability = \frac{\sum_{i=1}^n Treg_i}{n Tuptime}$ <p>where:</p> <p>Treg: is the time that a particular node is registered in the subnetwork since the last reset</p> <p>n: is the number of different nodes registered at least once since the last reset</p> <p>Tuptime: is the time since the last reset.</p> <p>For each of the 60 values corresponding to each period:</p> $Availability = \frac{\sum_{i=1}^n Treg_i}{n T}$ <p>where:</p> <p>Treg: is the time that a particular node is registered in the subnetwork during the corresponding period</p> <p>n: is the number of different nodes registered at least once in the corresponding period</p> <p>T: is the period value.</p> |
| Number of Nodes used in the Availability calculation | Integer16 | The Number of Nodes (n) associated to Average Availability                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           |

6398

6399 **N.2 Advanced Requirements for Base Nodes**6400 **N.2.1 Topology**

6401 The following object is a table including the list with the instantaneous “image” of the subnetwork topology.

6402 The last three values are calculated since the last reset of the base node.

6403 List of nodes with the following information of each of them:

| Object Name | Description   |       |  |
|-------------|---------------|-------|--|
| Topology    | Entry Element | Type  |  |
|             | MAC           | EUI48 |  |

|  |                |                                                                                      |
|--|----------------|--------------------------------------------------------------------------------------|
|  | State          | Integer8<br>(0:Disconnected;1:Terminal;2:Switch;3:Base)                              |
|  | LNID           | Integer16                                                                            |
|  | SID            | Integer8 (if not directly connected to the BN)                                       |
|  | LSID           | Integer8 (if it is a Switch)                                                         |
|  | Availability   | Integer32 (Total Time, in seconds, in which the Service Node is in registered state) |
|  | Disconnections | Integer32                                                                            |
|  | Coverage       | Integer16                                                                            |

#### 6404 N.2.1.1 Topology extension for Multi-PHY nodes

6405 In addition to the lists reported in N.2.1, when Multi-PHY promotion is supported, the following list is also  
6406 supported. The last five values are calculated since the last reset of the base node.

| Object Name | Description   |                                                                                                           |
|-------------|---------------|-----------------------------------------------------------------------------------------------------------|
| TopologyMP  | Entry Element | Type                                                                                                      |
|             | MAC           | EUI48                                                                                                     |
|             | State         | Integer8<br>(0:Disconnected;1:Terminal;2:Switch;<br>3:Base)                                               |
|             | LNID          | Integer16                                                                                                 |
|             | SID           | Integer8 (if not directly connected to the BN)                                                            |
|             | LSID          | Integer8 (if it is a Switch associated to a promotion accomplished using the PRO_REQ_x messages)          |
|             | MPLSID        | Integer8 (if it is a Switch associated to a promotion accomplished using the PRO_REQ_x_MultiPHY messages) |
|             | PCH           | Integer16 (The medium and the band (PLC) or channel (RF) used by the node for its registration)           |
|             | MPPCH         | Integer16 (The medium and the band (PLC) or channel (RF) associated to MPLSID)                            |

|  |                                |                                                                                                                       |
|--|--------------------------------|-----------------------------------------------------------------------------------------------------------------------|
|  | Availability in PLC medium     | Integer32 (Total Time, in seconds, in which the Service Node is in registered state and connected through PLC medium) |
|  | Availability in RF medium      | Integer32 (Total Time, in seconds, in which the Service Node is in registered state and connected through RF medium)  |
|  | Disconnections from PLC medium | Integer32                                                                                                             |
|  | Disconnections from RF medium  | Integer32                                                                                                             |
|  | Coverage                       | Integer16                                                                                                             |

6407

6408 **N.2.2 Traffic Sniffer and Topology Logging**

6409 Apart from these read-only objects, the node should provide a method to activate the sniffer option. For this  
 6410 purpose, the following objects would be available for read-write operations.

| Object Name | Description                    |                                                           |
|-------------|--------------------------------|-----------------------------------------------------------|
| Sniffer     | Entry Element                  | Type                                                      |
|             | Enable Sniffer                 | Integer8 (0 or 1)                                         |
|             | Enable Sniffer Optional Fields | Integer8 (0 or 1)                                         |
|             | Enable Topology                | Integer8 (0 or 1)                                         |
|             | Destination Address            | IPAddress (128 bits, supporting IPv4 and IPv6 addressing) |
|             | Destination Port               | Integer32                                                 |

6411

6412 When the node receives a “1” value for the Enable Sniffer object, it should activate the sniffer option sending  
 6413 every MAC PDU encapsulated in TCP to the destination address and port indicated by the corresponding  
 6414 objects. If Enable Sniffer Optional Fields object is also set to 1, the information will include optional fields.

6415 When the node receives a “1” value for the Enable Topology object, it should activate the topology logging  
 6416 function sending every topology change encapsulated in TCP to the destination address-port indicated by the  
 6417 corresponding objects.

6418

## 6419 N.2.2.1 Extension for Multi-PHY Traffic Sniffer and Topology Logging

6420 In addition to the object in N.2.2, when Multi-PHY promotion is supported, the following object is also  
6421 supported.

| Object Name | Description                       |                                                              |
|-------------|-----------------------------------|--------------------------------------------------------------|
| SnifferMP   | Entry Element                     | Type                                                         |
|             | Enable Sniffer                    | Integer8 (0 or 1)                                            |
|             | Enable Sniffer<br>Optional Fields | Integer8 (0 or 1)                                            |
|             | Enable<br>TopologyMP              | Integer8 (0 or 1)                                            |
|             | Destination<br>Address            | IPAddress (128 bits, supporting<br>IPv4 and IPv6 addressing) |
|             | Destination<br>Port               | Integer32                                                    |
|             | Sniffer<br>medium                 | Integer8<br><br>(0: PLC-only, 1: RF-only, 2:<br>PLC+RF)      |

6422

When the node receives a “1” value for the Enable SnifferMP object, it should activate the sniffer option sending every MAC PDU encapsulated in TCP to the destination address and port indicated by the corresponding objects. The band (PLC) and/or channel (RF) sniffed will be the ones in which the node is operating. If Enable Sniffer Optional Fields object is also set to 1, the information will include optional fields.

6427 When the node receives a “1” value for the Enable TopologyMP object, it should activate the topology logging  
6428 function sending every topology change encapsulated in TCP to the destination address and port indicated  
6429 by the corresponding objects.

## Annex O (informative)

### Traffic Sniffer and Topology Logging Protocol Definition

The protocol described here is just a recommendation to better describe sniffer and topology custom messages. However, manufacturers could use a different protocols as long as it complies with the specification.

The protocol used to transmit the sniffed traffic is a simple binary protocol encoded in big endian. The protocol encapsulates messages just having a fixed message format that could allow carrying other information in the future.

The device (BN) will open a socket when:

- The Topology, Sniffer or both are enabled
- Or, The TopologyMP, SnifferMP or both are enabled.

The basic message has the following format:

|         |        |                |
|---------|--------|----------------|
| 4 bytes | 1 byte | 'Length' bytes |
| Length  | Type   | Payload        |

Length: is the length of the payload, the messages are encoded sequentially and multiplexed using this length.

Type: is the type of the message.

The message types that can be carried now are the following:

| Message type | Identifier |
|--------------|------------|
| MAC_TX       | 0          |
| MAC_RX       | 1          |
| Topology     | 2          |
| TopologyMP   | 3          |

#### O.1 MAC\_TX and MAC\_RX

For the messages MAC\_TX and MAC\_RX the payload will have the following format:

|              |               |          |     |     |                 |
|--------------|---------------|----------|-----|-----|-----------------|
| 4            | 5             | 2        | 2   | Len | -               |
| Time Counter | Date and Time | PHY INFO | Len | PDU | Optional fields |

Time Counter: is a time counter of 10 microseconds that overflows every 12hours approximately.

6451 Date and Time: Number of seconds since 00:00 (midnight) 1 January, 1970 GMT.

6452 PHY Info:

| Bits     | type           | Description                                                                                                                                                                                                                                                                                  |
|----------|----------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 15 (MSb) | OptionalFields | 1 if optional fields are present, 0 otherwise.                                                                                                                                                                                                                                               |
| 14-13    | Interface      | 0: other<br>1: RF<br>2: PLC<br>3: reserved for future use                                                                                                                                                                                                                                    |
| 12-10    | reserved       |                                                                                                                                                                                                                                                                                              |
| 9-7      | Encoding       | If Interface is "PLC":<br>The encoding of the PDU:<br>0 – DBPSK<br>1 – DQPSK<br>2 – D8PSK<br>3 – DBPSK_CC<br>4 – DQPSK_CC<br>5 – D8PSK_CC<br>6 – DBPSK_R<br>7 – DQPSK_R<br>If Interface is not "PLC":<br>0 (not valid)                                                                       |
| 6-4      | SNR            | If Interface is "RF" or "PLC":<br>For RX: Signal to Noise ratio of the received message as defined in PRIME standard:<br>0: $\leq 0$ dB<br>1: $\leq 3$ dB<br>2: $\leq 6$ dB<br>...<br>7: $> 18$ dB<br>If Interface is "other" or "reserved for future use":<br>0 (not valid)<br>For TX:<br>0 |

|            |       |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             |
|------------|-------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 3-0 (LSbs) | Power | <p>If Interface is "PLC": 6453</p> <p>For RX: Reception power of the received message as defined in PRIME standard:</p> <p>0: <math>\leq 70</math> dBuV<br/> 1: <math>\leq 72</math> dBuV<br/> 2: <math>\leq 74</math> dBuV<br/> ...<br/> 15: <math>&gt; 98</math> dBuV</p> <p>For TX:</p> <p>0: Maximal output level (MOL) 1: MOL -3 dB<br/> 2: MOL -6 dB<br/> ...<br/> 7: MOL -21 dB</p> <p>If Interface is "RF":</p> <p>For RX: Reception power of the received message as defined in PRIME standard:</p> <p>0: <math>\leq -87</math> dBm<br/> 1: <math>\leq -82</math> dBm<br/> 2: <math>\leq -77</math> dBm<br/> ...<br/> 15: <math>&gt; -17</math> dBm</p> <p>For TX: Transmission power of the transmitted message as defined in PRIME standard:</p> <p>0: Maximal output level (MOL) 1: MOL -3 dB<br/> 2: MOL -6 dB<br/> ...<br/> 7: MOL -21 dB</p> <p>If Interface is "other" or "reserved for future use":<br/> 0 (not valid)</p> |
|------------|-------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|

6454 Len: the length of the PDU payload.

6455 PDU: the MAC PDU buffer.

6456 Optional fields: Optional information of each MAC PDU; this field can contain several fields with the following  
6457 format.

|          |              |              |
|----------|--------------|--------------|
| 1        | 1            | Field length |
| Field id | Field length | Data         |

6458 Some pre-defined optional fields are the following:

| Field id | length | type              |             |            | Description                                                                                                                                             |
|----------|--------|-------------------|-------------|------------|---------------------------------------------------------------------------------------------------------------------------------------------------------|
| 0        | 4      | PHY RX<br>Details | SNR         | 2<br>Bytes | Signed Integer indicating the Signal to Noise ratio in which this message was received in 0.1 dB steps.                                                 |
|          |        |                   | RX<br>Power | 2<br>bytes | Signed Integer indicating the Reception power in which this messages was received in 0.1 dBuV steps (PLC interface) or in 0.1 dBm steps (RF interface). |



|     |   |                 |                                                                                                                                                                                                                                 |
|-----|---|-----------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 1   | * | IFACE           | Text representation of the interface this messages was received/transmitted:<br>e.g: plc1, plc3 , rf.                                                                                                                           |
| 2   | 6 | EUI-48          | MAC Address of the service node origin of the PDU if uplink and destination if downlink.                                                                                                                                        |
| 3   | 4 | Duration        | The duration of the PDU in 10s of microseconds.                                                                                                                                                                                 |
| 255 | * | Vendor Specific | Content of this field is vendor specific, two first bytes in data shall contain the AppVendorId. It is recommended that vendors follow similar field structure with Field Id, Length and Data inside the Vendor Specific field. |

6459

6460 **O.2 Topology**

6461 For the Topology messages, the payload will have the following format:

|              |               |        |     |      |       |      |
|--------------|---------------|--------|-----|------|-------|------|
| 4            | 5             | 6      | 1   | 2    | 1     | 1    |
| Time Counter | Date and Time | EUI-48 | SID | LNID | STATE | LSID |

6462 Time Counter: is a time counter of 10 microseconds that overflows every 12 hours approximately.

6463 Date and Time: Number of seconds since 00:00 (midnight) 1 January, 1970 GMT.

6464 EUI-48: MAC Address of the service node.

6465 SID: Switch Identifier of the parent. 0 if directly connected to the Base Node.

6466 LNID: LNID of the service node.

6467 STATE: State of the service node as defined in PRIME Specification (0:Disconnected, 1:Terminal, 2:Switch,  
6468 3:Base).

6469 LSID: Switch Identifier if the service node is in Switch State.

6470 **O.3 TopologyMP**6471 For the TopologyMP messages in case where the Multi-PHY promotion is supported, the payload will have  
6472 the following format:

|   |   |   |   |   |   |   |   |   |
|---|---|---|---|---|---|---|---|---|
| 4 | 5 | 6 | 1 | 2 | 1 | 1 | 1 | 2 |
|---|---|---|---|---|---|---|---|---|

| Time Counter | Date and Time | EUI-48 | SID | LNID | STATE | LSID | MPLSID | PCH |
|--------------|---------------|--------|-----|------|-------|------|--------|-----|
|--------------|---------------|--------|-----|------|-------|------|--------|-----|

- 6473
- Time Counter: is a time counter of 10 microseconds that overflows every 12 hours approximately.
- 6474
- Date and Time: Number of seconds since 00:00 (midnight) 1 January, 1970 GMT.
- 6475
- EUI-48: MAC Address of the service node.
- 6476
- SID: Switch Identifier of the parent. 0 if directly connected to the Base Node.
- 6477
- LNID: LNID of the service node.
- 6478
- STATE: State of the service node as defined in PRIME Specification (0:Disconnected, 1:Terminal, 2:Switch,
- 6479
- 3:Base).
- 6480
- LSID: Switch Identifier if the service node is in Switch State.
- 6481
- MPLSID: Switch Identifier if the service node is in Switch State in specific medium and promoted to another
- 6482
- medium.
- 6483
- PCH: The medium and the band (PLC) or channel (RF) used by the node for its registration. This field is coded
- 6484
- as the PRO.PCH field of Table 28 of 4.4.2.6.5.1.
- 6485

## Annex P (normative)

### Node and network coverage calculation

#### P.1 Calculation of EntryCoverage

The node coverage, as indicated in the *EntryCoverage* field of the *macListExtDevicesInfo* attribute, shall be determined as follows:

##### P.1.1 When using PRIME 1.3.6 Alive

The coverage is defined as follows:

1. Upon the creation of a new node entry in the *macListExtDevicesInfo* attribute, the *EntryCoverage* field shall be initialized at 50% (5000):

$$EntryCoverage[0] = 5000$$

2. The *EntryCoverage* field shall be updated after each new ALV process directed to the node using the following formula:

$$EntryCoverage[n+1] = S * 10000 * (1 - alv\_fail) + (1 - S) * EntryCoverage[n]$$

Where:

- *EntryCoverage[n+1]* is the new coverage value.
  - *EntryCoverage[n]* is the previous coverage value.
  - *S* is a value between 0 and 1, calculated as per the *macEMASmoothing* attribute:  $S = 1 / (2^{macEMASmoothing})$ .
  - *alv\_fail* is 1 if the alive process fails (ALV\_S not received by Base Node), and 0 if it is successful (ALV\_S received by Base Node).
3. For nodes in Disconnected state, the last *EntryCoverage* value will remain stored for informative purposes.
  4. When a disconnected node is registered again, the algorithm restarts from point 1.

##### P.1.2 When using PRIME 1.4 Alive:

1. Upon creation of a new node entry in the *macListExtDevicesInfo* attribute, the *EntryCoverage* field shall be initialized taking into account the coverage of the parent node using the following formula:

$$EntryCoverageLatestLink[0] = 5000$$

$$EntryCoverage[0] = EntryCoverageParent * EntryCoverageLatestLink[0] / 10000$$

The *EntryCoverageParent* value for the base node is fixed at 10000

2. The *EntryCoverage* field shall be updated after each new ALV process in which the node is involved as intermediate or destination node using the following formula:

$$EntryCoverageLatestLink[n+1] = S * 10000 / (1 + alv\_rep\_count\_latest\_link) + (1 - S) * EntryCoverageLatestLink[n]$$

6520  $EntryCoverage[n+1] = EntryCoverageParent[n+1] * EntryCoverageLatestLink[n+1] / 10000$

6521 Where:

- 6522 • *EntryCoverageLatestLink[n+1]* is the new coverage of the latest link.
- 6523 • *EntryCoverageLatestLink[n]* is the previous coverage of the latest link.
- 6524 • S is a value between 0 and 1, calculated as per the *macEMASmoothing* attribute:  $S = 1 /$
- 6525  $(2^{macEMASmoothing})$ .
- 6526 • *EntryCoverage[n+1]* is the new coverage value.
- 6527 • *EntryCoverageParent[n+1]* is the new coverage value of the parent node.
- 6528 • *alv\_rep\_count\_latest\_link* is the total number of repetitions required in the last alive process
- 6529 for the final hop (uplink and downlink) for a particular node.
- 6530 • For intermediate switches, *alv\_rep\_count\_latest\_link* is obtained by summing the values of
- 6531 the REP\_U and REP\_D fields for the corresponding hop. The configuration of ALV.MIN\_LEVEL
- 6532 must be considered for this calculation: if ALV.MIN\_LEVEL > 0, all repetitions will be assigned
- 6533 to level 0 and no repetitions will be assigned to the other accumulated levels.
- 6534 • If the total number of repetitions exceeds the maximum repetitions value,  $(S * 10000) / (1 +$
- 6535 *alv\_rep\_count\_latest\_link*) is computed as 0.
- 6536 3. For nodes in Disconnected state, the last *EntryCoverage* value will remain stored for informative
- 6537 purposes.
- 6538 4. When a disconnected node is registered again, the algorithm restarts from point 1.

## 6539 P.2 Calculation of NetworkCoverage

6540 Network coverage, as included in the *macNetworkCoverage* attribute, shall be calculated as the average of

6541 the *EntryCoverage* fields of all the nodes in the *macListExtDevicesInfo* attribute. Nodes in Terminal and Switch

6542 states will use the *EntryCoverage* value, while nodes in Disconnected state will be computed as having an

6543 *EntryCoverage* of 0.

6544 Network coverage shall be updated every time an *EntryCoverage* field is updated.

## Annex Q (informative)

### TCP transport layer for DLMS: extension for optimal multiplexing 4-32 connections

The DLMS over TCP extension described in this annex is just a recommendation to optimize communication. However, manufacturers could use a different extension as long as it complies with the specification.

#### Q.1 Introduction

The DLMS/TCP implementation of both the base node (BN) and the data concentrator (DC) largely follows the standard DLMS/TCP specifications, as described in the DLMS UA 1000-2 Ed.12 of the Green Book in chapter 7.2 [30].

The annex describes the additional proprietary messages that have been added to improve the efficiency of the protocol to multiplex multiple devices over one connection.

#### Q.2 DLMS/TCP

##### Q.2.1 TCP configuration

The BN shall act as a TCP server and shall listen for connections on port 4059. The DC will act as TCP client and shall open a connection to the BN. The recommended TCP configurations for each one are as follows:

- BN (TCP Server):
  - Enable address reuse: Set the enable address reuse socket option to allow the server to bind to an address that may still be in the time-wait state.
  - Enable KeepAlive: Enable KeepAlive to detect dead client connections. The recommended KeepAlive parameters are as follows:
    - Idle Time: Set to 30 seconds before starting keep-alive probes.
    - Probe Count: Set to 10 probes before considering the connection dead.
    - Probe Interval: Set to 3 seconds between each probe.
- DC (TCP Client):
  - Set non-blocking mode: Configure the socket to operate in non-blocking mode to allow the application to remain responsive during connection attempts.
  - Enable KeepAlive: Enable KeepAlive to detect dead server connections. The recommended KeepAlive parameters are as follows:
    - Idle Time: Set to 30 seconds before starting keep-alive probes.
    - Probe Count: Set to 10 probes before considering the connection dead.
    - Probe Interval: Set to 3 seconds between each probe.

##### Q.2.2 DLMS over TCP standard protocol

There is only one connection that handles all the devices that are connected through the BN, the multiplexing of the devices is performed using the TCP wrapper via port fields.

The message format of DLMS/TCP as described in the green book is the following:

|         |             |           |         |           |
|---------|-------------|-----------|---------|-----------|
| 2 bytes | 2 bytes     | 2 bytes   | 2 bytes | N bytes   |
| Version | source port | dest port | length  | DLMS data |

6582 Where:

- 6583 • Version: 0x0001
- 6584 • Source port:
  - 6585 ○ Association ID for the data going DC → BN
  - 6586 ○ Device ID for the data going BN → DC
- 6587 • Destination port:
  - 6588 ○ Device ID for the data going DC → BN
  - 6589 ○ Association ID for the data going BN → DC
- 6590 • Length: Is the length of the *DLMS data* field
- 6591 • DLMS data: the data to be sent to the device

### 6592 Q.2.3 Reserved ports

6593 DLMS/TCP has a set of values for the ports that are reserved (green book 7.4.3.4):

| Client side reserved addresses |                     |
|--------------------------------|---------------------|
|                                | Wrapper Port Number |
| No-station                     | 0x0000              |
| Client Management Process      | 0x0001              |
| Public Client                  | 0x0010              |
| Server side reserved addresses |                     |
|                                | Wrapper Port Number |
| No-station                     | 0x0000              |
| Management Logical Device      | 0x0001              |
| Reserved for future use        | 0x0002...0x000F     |
| All-station (Broadcast)        | 0x007F              |

6594

### 6595 Q.2.4 DLMS/TCP standard compatibility

6596 If the DC does not send any custom message the BN will act as a standard bridge between DLMS/TCP and  
 6597 DLMS/432 multiplexing the devices via Device ID field. The Device ID field is a unique identifier assigned by  
 6598 the BN to each service node (SN) when a 4-32 Layer connection is opened. Device ID value 0x0000 (No-station)  
 6599 is reserved for custom messages. How the BN selects the Device ID for each SN is up to the implementer.  
 6600 However, it is recommended to start after the last reserved address (0x0080).

## 6601 Q.3 DLMS/TCP custom messages

### 6602 Q.3.1 Functionality of the custom messages

6603 The custom messages are meant to perform two tasks:

- 6604 • Improve the efficiency of the DLMS/TCP device discovery process.
- 6605 • Fix some corner cases that are not clear in the standard.

### 6606 Q.3.2 Addressing of the custom messages

6607 The used address for these messages is the Client management process in the client side and the No-station  
 6608 in the server side, or the same thing talking in other terms: we use the association (0x0001) in the no-station  
 6609 (0x0000) device for all the custom messages.

6610 **Q.3.3 Custom message definition**

6611 All the messages use the first byte to define the type of the message.

|        |         |
|--------|---------|
| 1 byte | N bytes |
| Type   | Payload |

6612

6613 The definitions below do not include the DLMS/TCP header.

6614 DC → BN:

| version (2 bytes) | Association ID (2 bytes) | Device ID (2 bytes) | Length (2 bytes) |
|-------------------|--------------------------|---------------------|------------------|
| 0x0001            | 0x0001                   | 0x0000              | <len>            |

6615

6616 BN → DC:

| version (2 bytes) | Device ID (2 bytes) | Association ID (2 bytes) | Length (2 bytes) |
|-------------------|---------------------|--------------------------|------------------|
| 0x0001            | 0x0000              | 0x0001                   | <len>            |

6617

6618 The type field will be included in the definition (the first byte) for clarity.

6619 **Q.3.3.1 BN → DC**

6620 **Q.3.3.1.1 NEW\_DEVICE\_NOTIFICATION (0x01)**

6621 Message to notify that a new SN has opened a 4-32 Layer connection with the BN.

6622 The Device ID in this message will be the key of the device and how this device will be addressed in all the  
6623 DLMS/TCP communications with the BN.

| Field          | Length (bytes) | Description                                                                                                                                                 |
|----------------|----------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Type (0x01)    | 1              | Type of message                                                                                                                                             |
| Device ID      | 2              | Device identifier that will be used to address the device this notification refers to, both in custom messages and in the standard DLMS/TCP communications. |
| Capabilities   | 2              | Flags with the capabilities of the device: <ul style="list-style-type: none"> <li>0x0001 : ARQ</li> </ul>                                                   |
| DLMS ID length | 1              | Length of the DLMS identifier                                                                                                                               |
| DLMS ID        | DLMS_ID_len    | DLMS Identifier of the reported device                                                                                                                      |
| EUI48          | 6              | EUI48 of the reported device                                                                                                                                |

6624

6625 **Q.3.3.1.2 REMOVE\_DEVICE\_NOTIFICATION (0x02)**

6626 This message is sent by the Base Node to the DC whenever a 4-32 sublayer connection with a service node is  
6627 closed. This may occur for various reasons, such as a prior Deregistration or Deletion request, or due to the  
6628 connection being closed spontaneously.

| Field       | Length (bytes) | Description                                         |
|-------------|----------------|-----------------------------------------------------|
| Type (0x02) | 1              | Type of message                                     |
| Device ID   | 2              | Identifier of the device that has been disconnected |

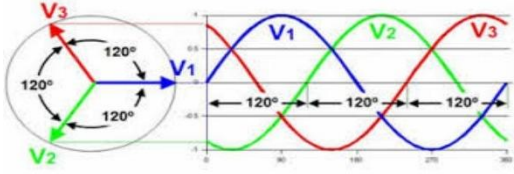
6629

6630 **Q.3.3.1.3 PHASE\_RESULT (0x08)**

6631 Result of the phase detection mechanism as an answer of the PHASE\_REQUEST message. Note that in the  
 6632 case of the phase detection mechanism the Device ID is not used, this is due to the phase detection relying  
 6633 on the Management Layer instead of the 4-32 Layer.

| Field       | Length (bytes) | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |
|-------------|----------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Type (0x08) | 1              | Type of message                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |
| EUI48       | 6              | EUI48 of the PRIME node whose phase detection has been requested.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |
| Result      | 1              | Result of the operation: <ul style="list-style-type: none"> <li>• 0: Successful phase detection</li> <li>• 1: Failure due to communication problem: The AppListZCStatus from the SN cannot be read.</li> <li>• 2: Failure due to invalid time reference from SN: TREF.TIME field from the SN set to invalid (0x80000000)</li> <li>• 3: Failure due to invalid time reference in BN: Internal error in the BN time reference</li> <li>• 4: Failure due to other reasons: Other errors (e.g. ZCStatus malformed, TREF header missing,...)</li> </ul> |
| Terminal BN | 1              | Terminal of the base node whose zero-crossing has been used (as indicated in the ZCStatus field in section 6.2.4): <ul style="list-style-type: none"> <li>• 0: invalid</li> <li>• 1: terminal block 1 (if single phase device only this will apply)</li> <li>• 2: terminal block 2</li> <li>• 3: terminal block 3</li> </ul>                                                                                                                                                                                                                       |
| Terminal SN | 1              | Terminal of the service node whose zero-crossing has been used (as indicated in the ZCStatus field in section 6.2.4): <ul style="list-style-type: none"> <li>• 0: invalid</li> <li>• 1: terminal block 1 (if single phase device only this will apply)</li> <li>• 2: terminal block 2</li> <li>• 3: terminal block 3</li> </ul>                                                                                                                                                                                                                    |



|             |   |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |
|-------------|---|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Phase Shift | 2 | <p>The phase angle of the voltage signals between the terminals of the BN and the SN measured in hundredths of degrees. It is a number in the range of [0,35999].</p> <p>The direction of the phase shift will use the following rule: It will be in the range of [1,17999] if the phase shift indicates that the voltage signal of the terminal of the BN is delayed compared to the SN. It will be in the range of [18001,35999] if the voltage signal of the SN is delayed compared to the BN.</p> <p><u>Example:</u> For the situation with the voltages as indicated in the following figure, in which the BN has its terminal (indicated in the field “Terminal BN”) connected to V1 and the SN has its terminal (indicated in the field “Terminal SN”) connected to V2, the field “Phase Shift” returned will be 24000 (considering an ideal communication scenario).</p>  |
|-------------|---|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|

6634

6635 **Q.3.3.2 DC → BN**6636 **Q.3.3.2.1 START\_REPORTING\_METERS (0x03)**

6637 Message to enable the notification of meter status report. Without this message the BN will not send any  
 6638 NEW\_DEVICE\_NOTIFICATION or REMOVE\_DEVICE\_NOTIFICATION.

| Field       | Length (bytes) | Description     |
|-------------|----------------|-----------------|
| Type (0x03) | 1              | Type of message |

6639

6640 **Q.3.3.2.2 DELETE\_METERS (0x04)**

6641 Message to force the disconnection of a meter from the DC. Upon the reception of this message the BN shall  
 6642 force the closing of the 4-32 Layer connection with the SN node indicated by the Device ID field by sending a  
 6643 CON\_CLS\_B message to the SN.

6644 Note: The manufacturer must take into account the highlighted note in section 5.5.6.2, in order to perform  
 6645 the remote closure of the CL-4-32 connection triggered by the BN.

| Field       | Length (bytes) | Description                                         |
|-------------|----------------|-----------------------------------------------------|
| Type (0x04) | 1              | Type of message                                     |
| Device ID   | 2              | Device Identifier of the device to be disconnected. |

6646

6647

**Q.3.3.2.3      ENABLE\_AUTO\_CLOSE (0x05)**

Enables the automatic close of the 4-32 Layer connection of the SN indicated by the Device ID field if the TCP connection is lost. If a SN has the auto-close enabled in the BN and the TCP connection is lost, the BN shall close the 4-32 Layer connection with the SN by sending a CON\_CLS\_B message. The purpose of this message is to make sure all the SNs are in a known state. This message is used before a DLMS association is opened with a device.

| Field       | Length (bytes) | Description                                                                   |
|-------------|----------------|-------------------------------------------------------------------------------|
| Type (0x05) | 1              | Type of message                                                               |
| Device ID   | 2              | Device Identifier of the device for which the automatic close will be enabled |

**Q.3.3.2.4      DISABLE\_AUTO\_CLOSE (0x06)**

Disables the automatic close of the 4-32 Layer connection of the SN indicated by the Device ID field. This message is sent after a DLMS association with a device is closed.

| Field       | Length (bytes) | Description                                                                    |
|-------------|----------------|--------------------------------------------------------------------------------|
| Type (0x06) | 1              | Type of message                                                                |
| Device ID   | 2              | Device Identifier of the device for which the automatic close will be disabled |

**Q.3.3.2.5      UNREGISTER\_METERS (0x07)**

Message to force the disconnection of a meter from the BN. Upon the reception of this message the BN shall force the PRIME disconnection of the SN node indicated by the Device ID field by sending a REG\_UNR\_B message to the SN.

| Field       | Length (bytes) | Description                                         |
|-------------|----------------|-----------------------------------------------------|
| Type (0x07) | 1              | Type of message                                     |
| Device ID   | 2              | Device Identifier of the device to be disconnected. |

**Q.3.3.2.6      PHASE\_REQUEST (0x09)**

Command to request the phase detection mechanism. Upon the reception of this message the BN node shall read the AppListZCStatus attribute (see section 6.2.4) of the SN indicated by EUI48 field.

| Field       | Length (bytes) | Description                                                       |
|-------------|----------------|-------------------------------------------------------------------|
| Type (0x09) | 1              | Type of message                                                   |
| EUI48       | 6              | EUI48 of the PRIME node whose phase detection is being requested. |

6668 **Q.4 Use cases**

6669 **Q.4.1 Service Node registration**

6670 The following Figure 187 illustrates the service node registration process. In this scenario, there are two  
6671 service nodes, one of which is already connected to the base node before the START\_REPORTING\_METERS  
6672 (0x03) message is sent. Upon receiving this message, the base node notifies the DC of all connected service  
6673 nodes, including those previously connected, using a NEW\_DEVICE\_NOTIFICATION message.

6674

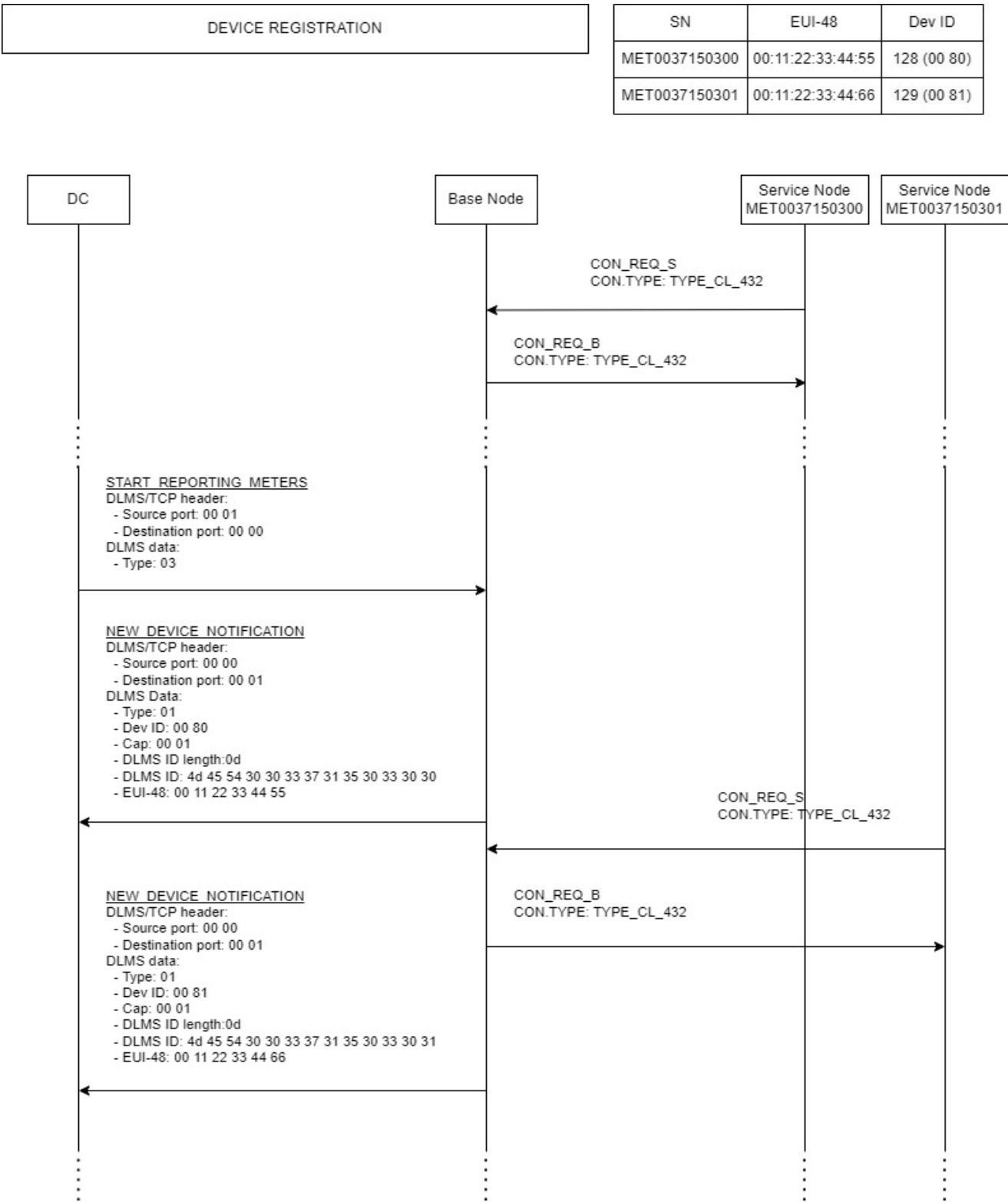


Figure 187 – Example of Service Node registration

**Q.4.2 DLMS communication**

The following Figure 188 illustrates the use case of DLMS communication between a DC and a service node. It assumes that the service node is already connected to the base node and registered with the DC, following a process such as the one described in section Q.4.1.

The ENABLE\_AUTO\_CLOSE message is sent before opening the DLMS association. Since TCP is a stream-oriented protocol, it is possible to send the ENABLE\_AUTO\_CLOSE message and the AARQ APDU within the same TCP segment. This reduces protocol overhead but depends on the specific implementation.

The DISABLE\_AUTO\_CLOSE message is sent when the DLMS association with a device is closed. In the example below, it is sent immediately after the DLMS RLRQ to prevent errors in cases where the RLRE is not received. As mentioned earlier, this message can also be sent within the same TCP segment as the RLRQ APDU.

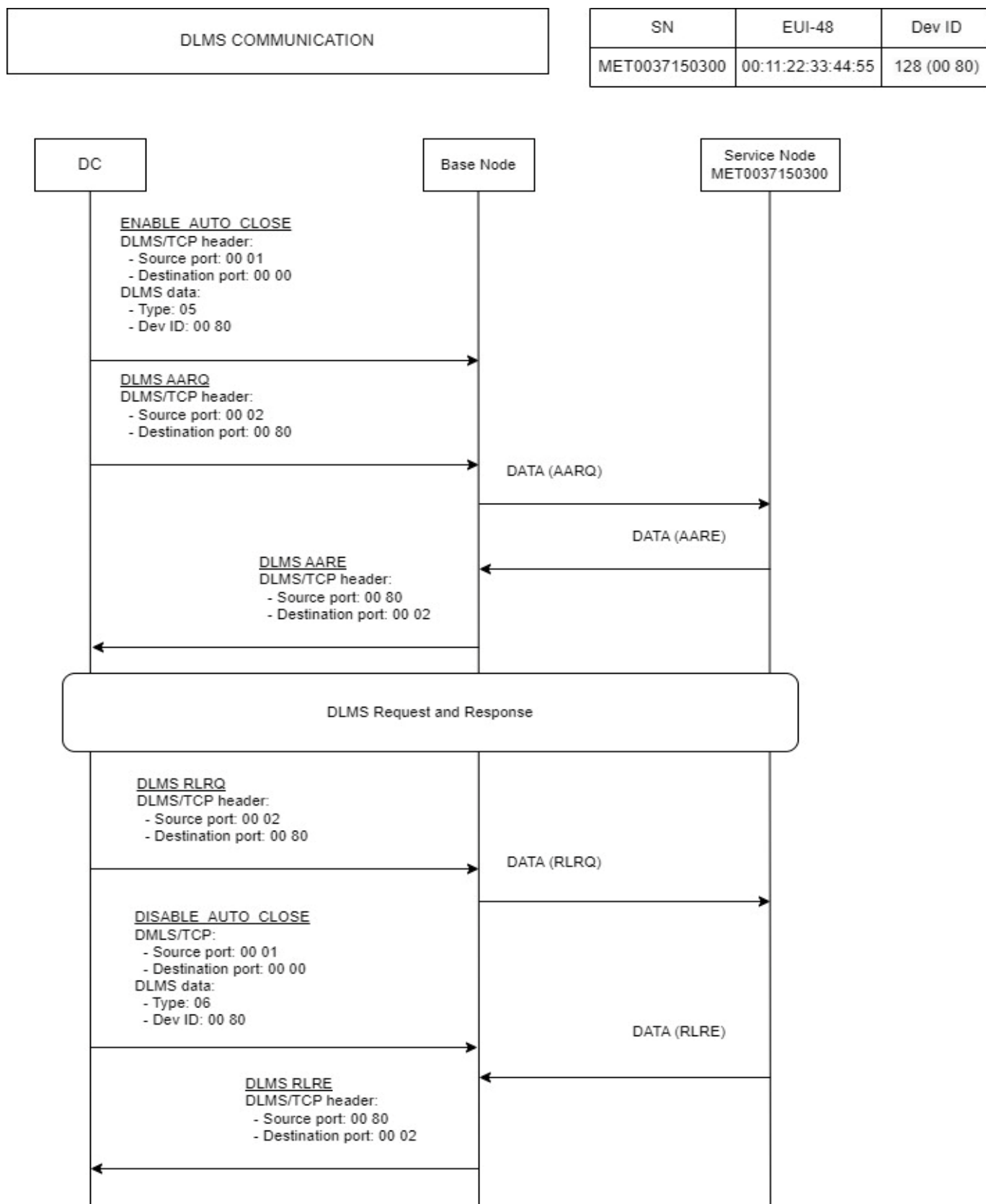


**Figure 188 – Example of DLMS communication**

---

6692 **Q.4.3 Service Node deletion and deregistration**

6693 In this Figure 189, the use of DELETE\_METERS and UNREGISTER\_METERS is demonstrated. The primary  
6694 difference between them is that the DELETE\_METERS message closes the 4-32 convergence layer connection  
6695 without deregistering at the PRIME level, whereas UNREGISTER\_METERS does imply deregistration at the  
6696 PRIME level. Since deregistration at the PRIME level automatically closes the 4-32 sublayer connection, there  
6697 is no need to close it explicitly.

6698 The base node should send a REMOVE\_DEVICE\_NOTIFICATION message each time a 4-32 connection with a  
6699 service node is closed.

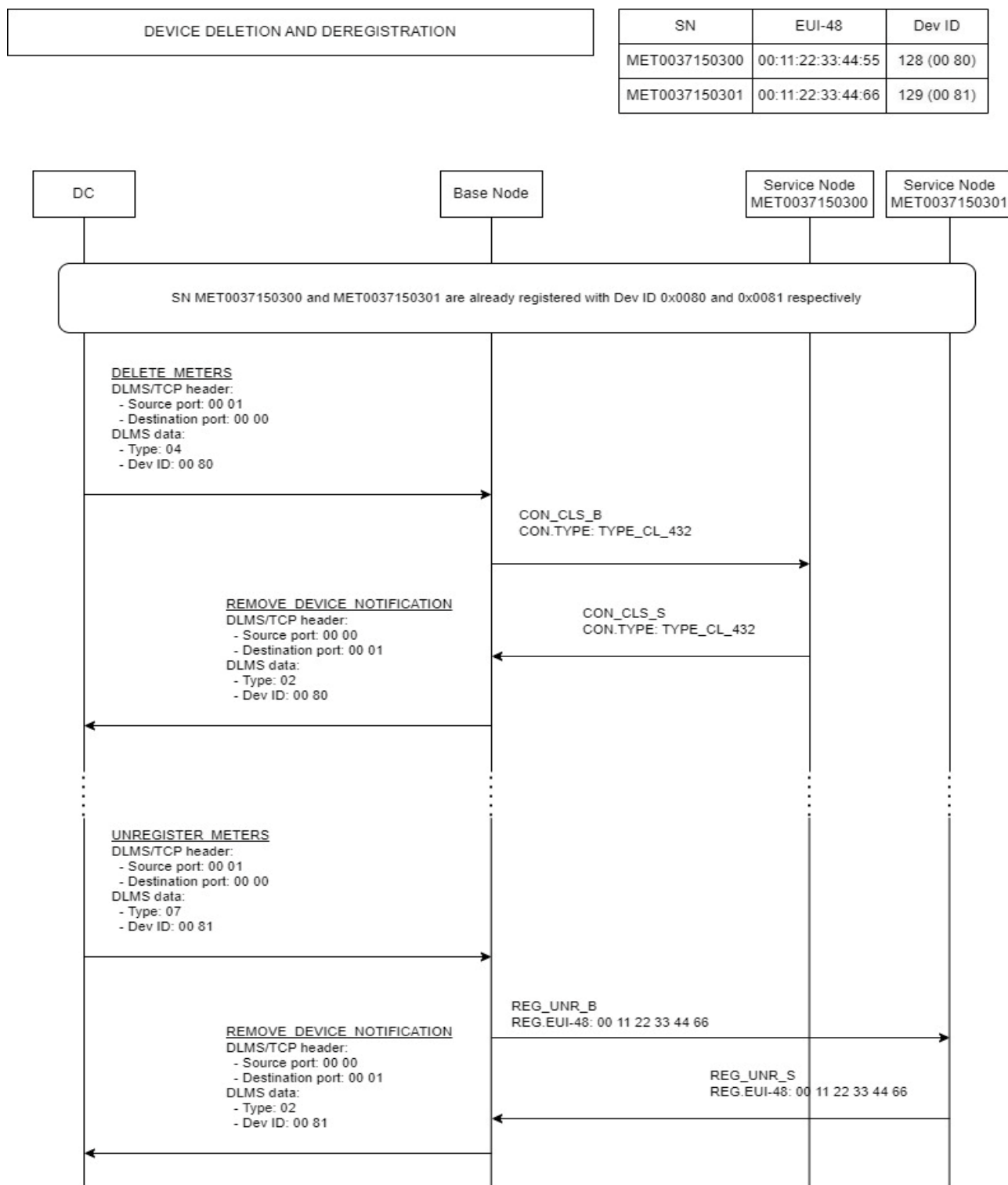


**Figure 189 – Example of Service Node deletion and deregistration**



Q.4.4 Phase detection

Figure 190 illustrates the use case for the phase detection query. The base node must retrieve the AppListZCStatus PIB attribute from the service node. Using this attribute, along with the information provided in the TREF subheader, the base node calculates the data to be included in the Phase Result message.

As described in sections Q.3.3.1.3 and Q.3.3.2.6, the phase detection mechanism operates at the management layer. This is because the AppListZCStatus PIB attribute must be accessed from the service nodes. Consequently, a management connection is required to perform the request. Whether this connection is kept open persistently or established on a per-request basis is left to the implementer’s decision.

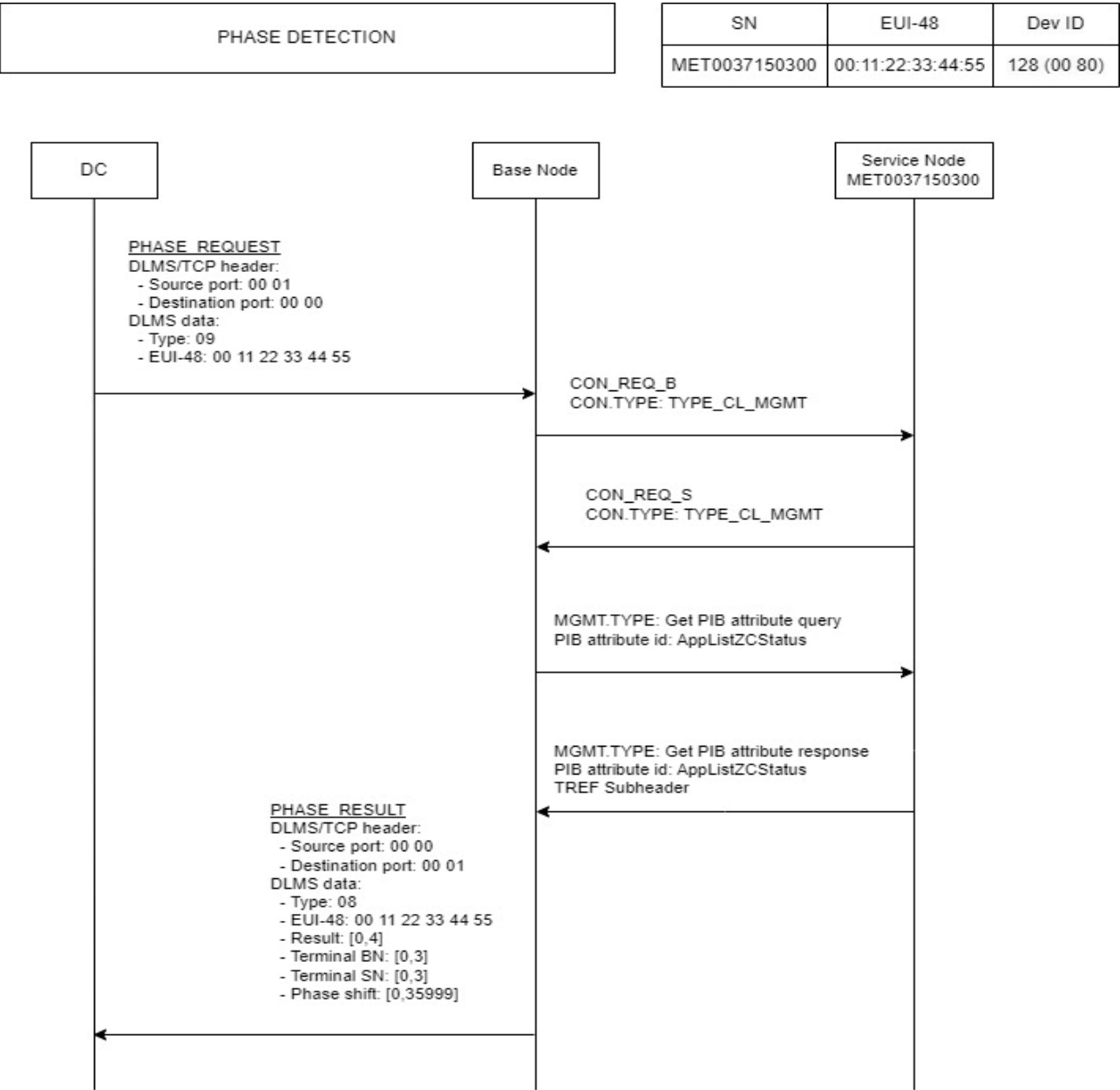


Figure 190 – Example of Phase detection request

**6716 List of authors (by alphabetical order)**

- 6717 *Ankou, Auguste (Itron)*
- 6718 *Arzuaga, Aitor (ZIV)*
- 6719 *Arzuaga, Txetxu (ZIV)*
- 6720 *Ballesteros, Miguel (Electrometer)*
- 6721 *Benedicto, Miguel Angel (Itron)*
- 6722 *Berganza, Inigo (Iberdrola)*
- 6723 *Bertoni, Guido (STMicroelectronics)*
- 6724 *Bisaglia, Paola (STMicroelectronics)*
- 6725 *Blasi, Danilo (STMicroelectronics)*
- 6726 *Bois, Simone (STMicroelectronics)*
- 6727 *Brunschweiler, Andreas (CURRENT Technologies International)*
- 6728 *Casone, Luca (STMicroelectronics)*
- 6729 *Cassin-Delauriere, Agnes (Texas Instruments)*
- 6730 *Du, Shu (Texas Instruments)*
- 6731 *Escrinhuela, Francisco (Ormazabal CURRENT)*
- 6732 *Estopiñan, Pedro (Atmel)*
- 6733 *Garai, Mikel (ZIV)*
- 6734 *Grasso, Riccardo (STMicroelectronics)*
- 6735 *Guerrieri, Lorenzo (STMicroelectronics)*
- 6736 *Jones, Kevin (Renesas)*
- 6737 *Kehn, Doug (Ormazabal CURRENT)*
- 6738 *Khemiri, Ibrahim (Sagemcom)*
- 6739 *Kim, Il Han (Texas Instruments)*
- 6740 *Lasciandare, Alessandro (STMicroelectronics)*
- 6741 *Liu, Weilin (CURRENT Technologies International)*
- 6742 *Llano, Asier (ZIV)*
- 6743 *Llorente, Isabel (Naturgy)*
- 6744 *Lunn, Andrew (CURRENT Technologies International)*
- 6745 *Manero, Eduardo (Microchip)*
- 6746 *Melguizo, Blanca (Microchip)*
- 6747 *Muñoz, Andrés (Atmel)*
- 6748 *Nasr, Imen (Sagemcom)*
- 6749 *Osorio, Xabier (ZIV)*
- 6750 *Piglione, Andrea (STMicroelectronics)*

---

|      |                                                               |
|------|---------------------------------------------------------------|
| 6751 | <i>Pulkkinen, Anssi (CURRENT Technologies International)</i>  |
| 6752 | <i>Rodriguez Roncero, Javier (Landis+Gyr)</i>                 |
| 6753 | <i>Romero, Gloria (Itron)</i>                                 |
| 6754 | <i>Saccani, Emile (STMicroelectronics)</i>                    |
| 6755 | <i>Salas, Jonay (Tecnalia)</i>                                |
| 6756 | <i>Sánchez, Agustín (Landis+Gyr)</i>                          |
| 6757 | <i>Sanz, Alfredo (Microchip)</i>                              |
| 6758 | <i>Sasaki, Yoshiyuki (Renesas)</i>                            |
| 6759 | <i>Scarpa, Vincenzo (STMicroelectronics)</i>                  |
| 6760 | <i>Schaub, Thomas (Landis+Gyr)</i>                            |
| 6761 | <i>Sedjai, Mohamed (CURRENT Technologies International)</i>   |
| 6762 | <i>Sendin, Alberto (Iberdrola)</i>                            |
| 6763 | <i>Sharma, Manu (CURRENT Technologies International)</i>      |
| 6764 | <i>Shibukawa, Akira (Renesas)</i>                             |
| 6765 | <i>Susella, Ruggero (STMicroelectronics)</i>                  |
| 6766 | <i>Tarruell, Frederic (Itron)</i>                             |
| 6767 | <i>Teijeiro, Jesús (Atmel)</i>                                |
| 6768 | <i>Treffiletti, Paolo (STMicroelectronics)</i>                |
| 6769 | <i>Varadarajan, Badri (Texas Instruments)</i>                 |
| 6770 | <i>Widmer, Hanspeter (CURRENT Technologies International)</i> |
| 6771 | <i>Wikiera, Jacek (CURRENT Technologies International)</i>    |