



1
2
3
4
5
6
7
8
9
10

ZigBee Specification

11
12
13
14
15
16

ZigBee Document 05-3474-21

August 5, 2015

Sponsored by: ZigBee Alliance

Accepted by

ZigBee Alliance Board of Directors

Abstract

The ZigBee Specification describes the infrastructure and services available to applications operating on the ZigBee platform.

Keywords

ZigBee, Stack, Network, Application, Profile, Framework, Device Description, Binding, Security

17

18

August 5, 2015

19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38

This page intentionally left blank.

39 **Notice of Use and Disclosure**

40 Copyright © ZigBee Alliance, Inc. (2015). All Rights Reserved. This information within this document is the property
41 of the ZigBee Alliance and its use and disclosure are restricted.

42 Elements of ZigBee Alliance specifications may be subject to third party intellectual property rights, including without
43 limitation, patent, copyright or trademark rights (such a third party may or may not be a member of ZigBee). ZigBee is
44 not responsible and shall not be held responsible in any manner for identifying or failing to identify any or all such
45 third party intellectual property rights.

46 This document and the information contained herein are provided on an “AS IS” basis and ZigBee DISCLAIMS ALL
47 WARRANTIES EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO (A) ANY WARRANTY THAT
48 THE USE OF THE INFORMATION HEREIN WILL NOT INFRINGE ANY RIGHTS OF THIRD PARTIES
49 (INCLUDING WITHOUT LIMITATION ANY INTELLECTUAL PROPERTY RIGHTS INCLUDING PATENT,
50 COPYRIGHT OR TRADEMARK RIGHTS) OR (B) ANY IMPLIED WARRANTIES OF MERCHANTABILITY,
51 FITNESS FOR A PARTICULAR PURPOSE, TITLE OR NONINFRINGEMENT. IN NO EVENT WILL ZIGBEE
52 BE LIABLE FOR ANY LOSS OF PROFITS, LOSS OF BUSINESS, LOSS OF USE OF DATA, INTERRUPTION
53 OF BUSINESS, OR FOR ANY OTHER DIRECT, INDIRECT, SPECIAL OR EXEMPLARY, INCIDENTAL,
54 PUNITIVE OR CONSEQUENTIAL DAMAGES OF ANY KIND, IN CONTRACT OR IN TORT, IN CONNEC-
55 TION WITH THIS DOCUMENT OR THE INFORMATION CONTAINED HEREIN, EVEN IF ADVISED OF THE
56 POSSIBILITY OF SUCH LOSS OR DAMAGE. All Company, brand and product names may be trademarks that are
57 the sole property of their respective owners.

58 The above notice and this paragraph must be included on all copies of this document that are made.

59 ZigBee Alliance, Inc.

60 508 Second Street
61 Suite 206
62 Davis, CA 95616
63 USA

64

65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84

This page intentionally left blank.

85 Document History

86 ZigBee Specification History

Revision	Date	Description
	December 14, 2004	ZigBee v.1.0 draft ratified
r06	February 17, 2006	ZigBee Specification (ZigBee document number 053474r06/07) incorporating errata and clarifications: ZigBee document numbers 053920r02, 053954r02, 06084r00, and 053474r07
r07	April 28, 2006	Changes made per Editorial comments on spreadsheet
r13	October 9, 2006	ZigBee-2006 Specification (see letter ballot comments and resolution in ZigBee document 064112)
r14	November 3, 2006	ZigBee-2007 Specification (adds features described in 064270, 064269, 064268, 064281, 064319, and 064293)
r15	December 12, 2006	ZigBee-2007 Specification incorporating errata and clarifications: 074746
r16	May 31, 2007	ZigBee-2007 Specification incorporating errata and clarifications: 07819
r17	October 19, 2007	ZigBee-2007 specification incorporating errata: 075318, 075053, 075164, 075098
r18	June 16, 2009	ZigBee-2007 specification incorporating errata: 08012
r19	September 28, 2010	ZigBee-2007 specification incorporating errata described in document 105413r04
r20	September 18, 2012	ZigBee-2007 specification incorporating errata described in 11-53778-r13 and 12-0030-01
r21	August 5, 2015	ZigBee specification incorporating large updates as follows: <ol style="list-style-type: none">1. Chapter 2 – Application Layer<ol style="list-style-type: none">a. Addition of Parent Announce ZDO messageb. Addition of over-the-air mechanism for detecting device's implemented specification version.2. Chapter 3 – Network Layer<ol style="list-style-type: none">a. Add End device timeout protocol and aging mechanism3. Chapter 4 – Security<ol style="list-style-type: none">a. Removal of High Securityb. Addition of Trust Center Link Key update servicesc. Cleanup of frame counter handling,d. Addition of Distributed Trust Center mode4. Annex D – MAC And PHY Sub-layer Clarifications<ol style="list-style-type: none">a. Update to 802.15.4-20115. Annex G – Inter-PAN

Revision	Date	Description
		<ul style="list-style-type: none">a. Formalization of Inter-PAN frame formats and service handling.6. Annex H – Inter-PAN Test Vectors<ul style="list-style-type: none">a. Addition of Green Power Inter-PAN test vectors.

87
88

89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108

This page intentionally left blank.

109

Table of Contents

110	Chapter 1	ZIGBEE PROTOCOL OVERVIEW	1
111	1.1	Protocol Description	1
112	1.1.1	Scope	1
113	1.1.2	Purpose	1
114	1.1.3	Stack Architecture	1
115	1.1.4	Network Topology	2
116	1.2	Conventions and Abbreviations	2
117	1.2.1	Symbols and Notation	2
118	1.2.2	Integers, Octets, and Their Representation	3
119	1.2.3	Transmission Order	3
120	1.2.4	Strings and String Operations	3
121	1.3	Acronyms and Abbreviations	3
122	1.4	Glossary	6
123	1.4.1	Definitions	6
124	1.5	References	11
125	1.5.1	ZigBee/IEEE References	11
126	1.5.2	Normative References	11
127	1.5.3	Informative References	12
128	CHAPTER 2	APPLICATION LAYER SPECIFICATION	14
129	2.1	General Description	14
130	2.1.1	Application Support Sub-Layer	14
131	2.1.2	Application Framework	14
132	2.1.3	ZigBee Device Objects	15
133	2.2	ZigBee Application Support (APS) Sub-Layer	15
134	2.2.1	Scope	15
135	2.2.2	Purpose	16
136	2.2.3	Application Support (APS) Sub-Layer Overview	16
137	2.2.4	Service Specification	17
138	2.2.5	Frame Formats	42
139	2.2.6	Command Frames	48
140	2.2.7	Constants and PIB Attributes	48
141	2.2.8	Functional Description	52
142	2.2.9	APS Sub-Layer Status Values	61
143	2.3	The ZigBee Application Framework	63
144	2.3.1	Creating a ZigBee Profile	63
145	2.3.2	ZigBee Descriptors	65
146	2.3.3	Functional Description	77
147	2.4	The ZigBee Device Profile	79
148	2.4.1	Scope	79
149	2.4.2	Device Profile Overview	79
150	2.4.3	Client Services	83
151	2.4.4	Server Services	128
152	2.4.5	ZDP Enumeration Description	184
153	2.4.6	Conformance	185
154	2.5	The ZigBee Device Objects (ZDO)	185
155	2.5.1	Scope	185
156	2.5.2	Device Object Descriptions	185
157	2.5.3	Layer Interface Description	190
158	2.5.4	System Usage	190
159	2.5.5	Configuration Attributes	211

160	Chapter 3 NETWORK SPECIFICATION	219
161	3.1 General Description.....	219
162	3.1.1 Network (NWK) Layer Overview.....	219
163	3.1.2 Network Layer Data Entity (NLDE).....	219
164	3.2 Service Specification	220
165	3.2.1 NWK Data Service.....	220
166	3.2.2 NWK Management Service	227
167	3.3 Frame Formats.....	263
168	3.3.1 General NPDU Frame Format	263
169	3.3.2 Format of Individual Frame Types	268
170	3.4 Command Frames.....	269
171	3.4.1 Route Request Command.....	270
172	3.4.2 Route Reply Command.....	273
173	3.4.3 Network Status Command	276
174	3.4.4 Leave Command	278
175	3.4.5 Route Record Command	280
176	3.4.6 Rejoin Request Command	281
177	3.4.7 Rejoin Response Command	282
178	3.4.8 Link Status Command.....	283
179	3.4.9 Network Report Command	285
180	3.4.10 Network Update Command.....	287
181	3.4.11 End Device Timeout Request Command	289
182	3.4.12 End Device Timeout Response Command.....	291
183	3.5 Constants and NIB Attributes.....	293
184	3.5.1 NWK Constants	293
185	3.5.2 NWK Information Base	294
186	3.6 Functional Description	302
187	3.6.1 Network and Device Maintenance	302
188	3.6.2 Transmission and Reception	334
189	3.6.3 Routing.....	337
190	3.6.4 Scheduling Beacon Transmissions.....	352
191	3.6.5 Broadcast Communication.....	354
192	3.6.6 Multicast Communication.....	357
193	3.6.7 NWK Information in the MAC Beacons.....	360
194	3.6.8 Persistent Data	362
195	3.6.9 Low Power Routers (LPR).....	362
196	3.6.10 End Device Aging and Management	363
197	3.7 NWK Layer Status Values	371
198	Chapter 4 SECURITY SERVICES SPECIFICATION	375
199	4.1 Document Organization.....	375
200	4.2 General Description.....	375
201	4.2.1 Security Architecture and Design	375
202	4.2.2 NWK Layer Security	378
203	4.2.3 APL Layer Security	379
204	4.2.4 Trust Center Role	380
205	4.3 NWK Layer Security.....	380
206	4.3.1 Frame Security	380
207	4.3.2 Secured NPDU Frame.....	383
208	4.3.3 Security-Related NIB Attributes	383
209	4.3.4 Network Frame Counter Requirements.....	385
210	4.4 APS Layer Security	386
211	4.4.1 Frame Security	387
212	4.4.2 Transport-Key Services.....	392
213	4.4.3 Update Device Services	398
214	4.4.4 Remove Device Services	400

215	4.4.5	Request Key Services.....	402
216	4.4.6	Switch Key Services	406
217	4.4.7	Verify-Key Services.....	409
218	4.4.8	Confirm-Key Services.....	412
219	4.4.9	Secured APDU Frame.....	415
220	4.4.10	Command Frames	416
221	4.4.11	Security-Related AIB Attributes	422
222	4.5	Common Security Elements	424
223	4.5.1	Auxiliary Frame Header Format	424
224	4.5.2	Security Parameters.....	426
225	4.5.3	Cryptographic Key Hierarchy	427
226	4.5.4	Implementation Requirements	427
227	4.6	Functional Description	428
228	4.6.1	ZigBee Security Initialization	428
229	4.6.2	Trust Center Application.....	428
230	4.6.3	Security Procedures.....	429
231	4.7	Security Operations in Centralized Security Networks	442
232	4.7.1	Trust Center Policies.....	442
233	4.7.2	Trust Center Link Keys.....	442
234	4.7.3	Trust Center Policy Values	442
235	4.8	Security Operations in Distributed Security Networks.....	451
236	4.8.1	Trust Center Address	451
237	4.8.2	Network Key Updates	451
238	4.8.3	Link Keys.....	451
239	4.8.4	Application Link Keys	452
240	4.8.5	Requesting Keys	452
241	4.9	Device Operations	452
242	4.9.1	Joining Device Policy Values	452
243	4.9.2	Trust Center Address	453
244	4.9.3	Trust Center Link Keys.....	453
245	4.9.4	Receiving new Link Keys	453
246	4.9.5	Requesting a Link Key.....	453
247	Annex A	CCM* MODE OF OPERATION.....	456
248	A.1	Notation and Representation	456
249	A.2	CCM* Mode Encryption and Authentication Transformation	456
250	A.2.1	Input Transformation	457
251	A.2.2	Authentication Transformation	457
252	A.2.3	Encryption Transformation	458
253	A.3	CCM* Mode Decryption and Authentication Checking Transformation.....	458
254	A.3.1	Decryption Transformation.....	458
255	A.3.2	Authentication Checking Transformation.....	459
256	A.4	Restrictions	459
257	Annex B	SECURITY BUILDING BLOCKS.....	460
258	B.1	Symmetric-Key Cryptographic Building Blocks.....	460
259	B.1.1	Block-Cipher.....	460
260	B.1.2	Mode of Operation.....	460
261	B.1.3	Cryptographic Hash Function	460
262	B.1.4	Keyed Hash Function for Message Authentication.....	461
263	B.1.5	Specialized Keyed Hash Function for Message Authentication	461
264	B.1.6	Challenge Domain Parameters	461
265	B.2	Key Agreement Schemes.....	461
266	B.2.1	Symmetric-Key Key Agreement Scheme	461
267	B.3	Challenge Domain Parameter Generation and Validation.....	462
268	B.3.1	Challenge Domain Parameter Generation.....	462

269	B.3.2 Challenge Domain Parameter Verification	462
270	B.4 Challenge Validation Primitive	462
271	B.5 Secret Key Generation (SKG) Primitive	463
272	B.6 Block-Cipher-Based Cryptographic Hash Function	463
273	B.7 Symmetric-Key Authenticated Key Agreement Scheme	465
274	B.7.1 Initiator Transformation	466
275	B.7.2 Responder Transformation	467
276	B.8 Mutual Symmetric-Key Entity Authentication	468
277	B.8.1 Initiator Transformation	469
278	B.8.2 Responder Transformation	470
279	Annex C TEST VECTORS FOR CRYPTOGRAPHIC BUILDING BLOCKS	472
280	C.1 Data Conversions	472
281	C.2 AES Block Cipher	472
282	C.3 CCM* Mode Encryption and Authentication Transformation	472
283	C.3.1 Input Transformation	473
284	C.3.2 Authentication Transformation	473
285	C.3.3 Encryption Transformation	474
286	C.4 CCM* Mode Decryption and Authentication Checking Transformation	475
287	C.4.1 Decryption Transformation	475
288	C.4.2 Authentication Checking Transformation	476
289	C.5 Cryptographic Hash Function	477
290	C.5.1 Test Vector Set 1	477
291	C.5.2 Test Vector Set 2	477
292	C.5.3 Test Vector Set 3	479
293	C.5.4 Test Vector 4	480
294	C.5.5 Test Vector 5	480
295	C.5.6 Test Vector 6	481
296	C.6 Keyed Hash Function for Message Authentication	482
297	C.6.1 Test Vector Set 1	482
298	C.6.2 Test Vector Set 2	483
299	C.6.3 Specialized Keyed Hash Function for Message Authentication	484
300	Annex D MAC AND PHY SUB-LAYER CLARIFICATIONS	486
301	D.1 Introduction	486
302	D.1.1 Scope	486
303	D.1.2 Purpose	486
304	D.2 Stack Size Issues	486
305	D.3 MAC Association	487
306	D.4 aMaxMACFrameSize	489
307	D.5 Frame Version Value	489
308	D.6 CSMA Backoff Timing	490
309	D.7 MAC Interface Changes	490
310	D.7.1 Additional Primitives accessed through the MLME-SAP	490
311	D.7.2 MLME-POLL.indication	490
312	Annex E OPERATING NETWORK MANAGER AS NETWORK CHANNEL MANAGER FOR INTERFERENCE REPORTING AND RESOLUTION	493
314	Annex F USAGE OF MULTIPLE FREQUENCY BANDS	495
315	F.1 Introduction	495
316	F.1.1 Scope	495
317	F.1.2 Purpose	495
318	F.2 Channels and Channel Masks Management General Guideline	495
319	F.2.1 Channel Selection During Network Establishment	495
320	F.2.2 The Frequency Agility Feature Related Points	496

321	F.2.3 Network Management Services and Client Services Affected by Multiple Frequency Bands	
322	Support	496
323	F.3 Timing Issues	496
324	Annex G Inter-PAN COMMUNICATIONS.....	499
325	G.1 Scope and Purpose.....	499
326	G.2 General Description.....	499
327	G.2.1 What Inter-PAN APS Does.....	499
328	G.2.2 Service Specification.....	500
329	G.2.3 The INTRP-DATA.request Primitive.....	501
330	G.2.4 The GP-DATA.request Primitive.....	503
331	G.2.5 The INTRP-DATA.confirm Primitive	506
332	G.2.6 The GP-DATA.confirm Primitive	506
333	G.2.7 The GP-SEC.request Primitive	507
334	G.2.8 The GP-SEC.response Primitive	509
335	G.2.9 The INTRP-DATA.indication Primitive	512
336	G.2.10 The GP-DATA.indication Primitive	515
337	G.2.11 Qualifying and Testing of Inter-PAN Messages	519
338	G.3 Frame Formats.....	519
339	G.3.1 MAC Header.....	520
340	G.3.2 Network Header.....	521
341	G.3.3 Inter-PAN APS Header.....	524
342	G.4 Frame Processing.....	525
343	G.4.1 Inter-PAN Transmission (non Green Power Device Frames).....	525
344	G.4.2 Inter-PAN Reception (non Green Power Device Frames)	526
345	G.4.3 Green Power Device Frame Transmission.....	527
346	G.4.4 Green Power Device Frame Reception	527
347	G.5 Green Power Security Stub Operations	528
348	G.5.1 Per GPDF Security Level and Key Selection.....	528
349	G.5.2 Construction of AES Nonce	528
350	G.5.3 Initialization	530
351	G.5.4 Outgoing frame encryption and authentication	530
352	G.5.5 Incoming frame decryption and authentication check.....	531
353	G.5.6 Reporting to the next higher layer.....	532
354	G.6 Inter-PAN Best Practices.....	532
355	Annex H SECURITY TEST VECTORS FOR GREEN POWER DEVICE FRAMES.....	534
356	H.1 Overview	534
357	H.2 Security Test Vectors for a Shared Key	534
358	H.2.1 Common Settings.....	534
359	H.2.2 SecurityLevel = 0b01	534
360	H.2.3 SecurityLevel = 0b10	536
361	H.2.4 SecurityLevel = 0b11	537
362	H.3 Security test vectors for an individual key	538
363	H.3.1 Common settings	538
364	H.3.2 SecurityLevel=0b01	539
365	H.3.3 SecurityLevel=0b10	539
366	H.3.4 SecurityLevel=0b11	539
367	H.4 Security test vectors for bidirectional operation	539
368	H.4.1 Common settings	539
369	H.4.2 Security test vectors for a shared key	540
370	H.4.3 Security test vectors for an individual key	541
371		
372		
373		
374		

375
376
377
378
379
380
381
382
383
384
385

386 List of Tables

387	Table 1.1 ZigBee Protocol Versions.....	7
388	Table 2.1 APSDE-SAP Primitives.....	17
389	Table 2.2 APSDE-DATA.request Parameters.....	18
390	Table 2.3 APSDE-DATA.confirm Parameters.....	22
391	Table 2.4 APSDE-DATA.indication Parameters.....	24
392	Table 2.5 Summary of the Primitives Accessed Through the APSME-SAP.....	27
393	Table 2.6 APSME-BIND.request Parameters.....	28
394	Table 2.7 APSME-BIND.confirm Parameters.....	29
395	Table 2.8 APSME-UNBIND.request Parameters.....	30
396	Table 2.9 APSME-UNBIND.confirm Parameters.....	32
397	Table 2.10 APSME-GET.request Parameters.....	33
398	Table 2.11 APSME-GET.confirm Parameters.....	34
399	Table 2.12 APSME-SET.request Parameters	35
400	Table 2.13 APSME-SET.confirm Parameters	36
401	Table 2.14 APSME-ADD-GROUP.request Parameters	37
402	Table 2.15 APSME-ADD-GROUP.confirm Parameters	38
403	Table 2.16 APSME-REMOVE-GROUP.request Parameters	39
404	Table 2.17 APSME-REMOVE-GROUP.confirm Parameters	40
405	Table 2.18 APSME-REMOVE-ALL-GROUPS.request Parameters	40
406	Table 2.19 APSME-REMOVE-ALL-GROUPS.confirm Parameters	41
407	Table 2.20 Values of the Frame Type Sub-Field	43
408	Table 2.21 Values of the Delivery Mode Sub-Field	43
409	Table 2.22 Values of the Fragmentation Sub-Field	45
410	Table 2.23 APS Sub-Layer Constants	48
411	Table 2.24 APS IB Attributes	49
412	Table 2.25 Group Table Entry Format.....	51
413	Table 2.26 <i>apsMaxWindowSize</i> by Endpoint Number.....	51
414	Table 2.27 APS Sub-layer Status Values.....	62
415	Table 2.28 ZigBee Descriptors	65
416	Table 2.29 Fields of the Node Descriptor	67
417	Table 2.30 Values of the Logical Type Field	68
418	Table 2.31 Values of the Frequency Band Field.....	68
419	Table 2.32 Server Mask Bit Assignments	70
420	Table 2.33 Descriptor Capability Bit Assignments	70
421	Table 2.34 Fields of the Node Power Descriptor.....	71
422	Table 2.35 Values of the Current Power Mode Field	71
423	Table 2.36 Values of the Available Power Sources Field.....	72
424	Table 2.37 Values of the Current Power Sources Field.....	72
425	Table 2.38 Values of the Current Power Source Level Field	73
426	Table 2.39 Fields of the Simple Descriptor	73
427	Table 2.40 Values of the Application Device Version Field	74
428	Table 2.41 Fields of the Complex Descriptor.....	75
429	Table 2.42 Values of the Character Set Identifier Sub-Field	76
430	Table 2.43 Fields of the User Descriptor.....	77
431	Table 2.44 Profile ID Endpoint Matching Rules	77
432	Table 2.45 Device and Service Discovery Client Services Commands.....	83
433	Table 2.46 Fields of the NWK_addr_req Command	85
434	Table 2.47 Fields of the IEEE_addr_req Command	86
435	Table 2.48 Fields of the Node_Desc_req Command	87
436	Table 2.49 Fields of the Power_Desc_req Command.....	88
437	Table 2.50 Fields of the Simple_Desc_req Command	89
438	Table 2.51 Fields of the Active_EP_req Command	89
439	Table 2.52 Fields of the Match_Desc_req Command.....	90
440	Table 2.53 Fields of the Complex_Desc_req Command	91

441	Table 2.54 Fields of the User_Desc_req Command	92
442	Table 2.55 Fields of the Discovery_Cache_req Command	93
443	Table 2.56 Fields of the Device_ance Command	93
444	Table 2.57 - Format of the ChildInfo Structure	94
445	Table 2.58 Fields of the User_Desc_set Command.....	96
446	Table 2.59 Fields of the System_Server_Discovery_req Command	97
447	Table 2.60 Fields of the Discovery_store_req Command.....	97
448	Table 2.61 Fields of the Node_Desc_store_req Command	99
449	Table 2.62 Fields of the Power_Desc_store_req Command	99
450	Table 2.63 Fields of the Active_EP_store_req Command.....	100
451	Table 2.64 Fields of the Simple_Desc_store_req Command.....	101
452	Table 2.65 Fields of the Remove_node_cache_req Command.....	102
453	Table 2.66 Fields of the Find_node_cache_req Command Frame	103
454	Table 2.67 Fields of the Extended_Simple_Desc_req Command	104
455	Table 2.68 Fields of the Extended_Active_EP_req Command	105
456	Table 2.69 End Device Bind, Bind, Unbind, and Bind Management Client Service Commands	106
457	Table 2.70 Fields of the End_Device_Bind_req Command	107
458	Table 2.71 Fields of the Bind_req Command.....	108
459	Table 2.72 Fields of the Unbind_req Command.....	110
460	Table 2.73 Fields of the Bind_Register_req Command.....	111
461	Table 2.74 Fields of the Replace_Device_req Command.....	112
462	Table 2.75 Fields of the Store_Bkup_Bind_Entry_req Command	113
463	Table 2.76 Fields of the Remove_Bkup_Bind_Entry_req Command	114
464	Table 2.77 Fields of the Backup_Bind_Table_req Command.....	115
465	Table 2.78 Fields of the Recover_Bind_Table_req Command.....	116
466	Table 2.79 Fields of the Backup_Source_Bind_req Command.....	117
467	Table 2.80 Fields of the Recover_Source_Bind_req Command.....	118
468	Table 2.81 Network Management Client Services Commands	119
469	Table 2.82 Fields of the Mgmt_NWK_Disc_req Command	119
470	Table 2.83 Fields of the Mgmt_Lqi_req Command	120
471	Table 2.84 Fields of the Mgmt_Rtg_req Command	121
472	Table 2.85 Fields of the Mgmt_Bind_req Command	122
473	Table 2.86 Fields of the Mgmt_Leave_req Command	123
474	Table 2.87 Fields of the Mgmt_Direct_Join_req Command	124
475	Table 2.88 Fields of the Mgmt_Permit_Joining_req Command.....	125
476	Table 2.89 Fields of the Mgmt_Cache_req Command	126
477	Table 2.90 Fields of the Mgmt_NWK_Update_req Command	127
478	Table 2.91 Device and Service Discovery Server Service Primitives	129
479	Table 2.92 Fields of the NWK_addr_rsp Command	130
480	Table 2.93 IEEE_addr_rsp Parameters	132
481	Table 2.94 Fields of the Node_Desc_rsp Command	134
482	Table 2.95 Fields of the Power_Desc_rsp Command.....	136
483	Table 2.96 Fields of the Simple_Desc_rsp Command.....	137
484	Table 2.97 Fields of the Active_EP_rsp Command.....	138
485	Table 2.98 Fields of the Match_Desc_rsp Command.....	140
486	Table 2.99 Fields of the Complex_Desc_rsp Command	142
487	Table 2.100 Fields of the User_Desc_rsp Command	144
488	Table 2.101 Fields of the System_Server_Discovery_rsp Command	146
489	Table 2.102 Fields of the User_Desc_conf Command	147
490	Table 2.103 Fields of the Discovery_Cache_rsp Command.....	148
491	Table 2.104 Fields of the Discovery_store_rsp Command.....	148
492	Table 2.105 Fields of the Node_Desc_store_rsp Command.....	149
493	Table 2.106 Fields of the Power_Desc_store_rsp Command	150
494	Table 2.107 Fields of the Active_EP_store_rsp Command	151
495	Table 2.108 Fields of the Simple_Desc_store_rsp Command	151
496	Table 2.109 Fields of the Remove_node_cache_rsp Command.....	152

497	Table 2.110 Fields of the Find_node_cache_rsp Command	153
498	Table 2.111 Fields of the Extended_Simple_Desc_rsp Command	154
499	Table 2.112 Fields of the Extended_Active_EP_rsp Command	156
500	Table 2.113 Fields of the Parent_ance_rsp	157
501	Table 2.114 End Device Bind, Unbind and Bind Management Server Services Primitives	158
502	Table 2.115 Fields of the End_Device_Bind_rsp Command	159
503	Table 2.116 Fields of the Bind_rsp Command	160
504	Table 2.117 Fields of the Unbind_rsp Command	161
505	Table 2.118 Fields of the Bind_Register_rsp Command	163
506	Table 2.119 Fields of the Replace_Device_rsp Command	164
507	Table 2.120 Fields of the Store_Bkup_Bind_Entry_rsp Command	164
508	Table 2.121 Fields of the Remove_Bkup_Bind_Entry_rsp Command	165
509	Table 2.122 Fields of the Backup_Bind_Table_rsp Command	166
510	Table 2.123 Fields of the Recover_Bind_Table_rsp Command	167
511	Table 2.124 Fields of the Backup_Source_Bind_rsp Command	168
512	Table 2.125 Fields of the Recover_Source_Bind_rsp Command	168
513	Table 2.126 Network Management Server Service Commands	169
514	Table 2.127 Fields of the Mgmt_NWK_Disc_rsp Command	170
515	Table 2.128 NetworkList Record Format	171
516	Table 2.129 Fields of the Mgmt_Lqi_rsp Command	172
517	Table 2.130 NeighborTableList Record Format	173
518	Table 2.131 Fields of the Mgmt_Rtg_rsp Command	175
519	Table 2.132 RoutingTableList Record Format	176
520	Table 2.133 Fields of the Mgmt_Bind_rsp Command	177
521	Table 2.134 BindingTableList Record Format	178
522	Table 2.135 Fields of the Mgmt_Leave_rsp Command	179
523	Table 2.136 Fields of the Mgmt_Direct_Join_rsp Command	180
524	Table 2.137 Fields of the Mgmt_Permit_Joining_rsp Command	181
525	Table 2.138 Fields of the Mgmt_Cache_rsp Command	181
526	Table 2.139 DiscoveryCacheList Record Format	182
527	Table 2.140 Fields of the Mgmt_NWK_Update_notify Command	183
528	Table 2.141 ZDP Enumerations Description	184
529	Table 2.142 ZigBee Device Objects	191
530	Table 2.143 Startup Attributes	202
531	Table 2.144 Additional Commissioning Attributes	203
532	Table 2.145 Device and Service Discovery Attributes	204
533	Table 2.146 Security Manager Attributes	206
534	Table 2.147 Binding Manager Attributes	208
535	Table 2.148 Network Manager Attributes	210
536	Table 2.149 Configuration Attributes	212
537	Table 2.150 Configuration Attribute Definitions	213
538	Table 3.1 NLDE-SAP Primitives	220
539	Table 3.2 NLDE-DATA.request Parameters	221
540	Table 3.3 NLDE-DATA.confirm Parameters	225
541	Table 3.4 NLDE-DATA.indication Parameters	226
542	Table 3.5 Summary of the Primitives Accessed Through the NLME-SAP	227
543	Table 3.6 NLME-NETWORK-DISCOVERY.request Parameters	229
544	Table 3.7 NLME-NETWORK-DISCOVERY.confirm Parameters	230
545	Table 3.8 Network Descriptor Information Fields	230
546	Table 3.9 NLME-NETWORK-FORMATION.request Parameters	232
547	Table 3.10 NLME-NETWORK-FORMATION.confirm Parameters	234
548	Table 3.11 NLME-PERMIT-JOINING.request Parameters	235
549	Table 3.12 NLME-PERMIT-JOINING.confirm Parameters	236
550	Table 3.13 NLME-START-ROUTER.request Parameters	236
551	Table 3.14 NLME-START-ROUTER.confirm Parameters	238
552	Table 3.15 NLME-ED-SCAN.request Parameters	238

553	Table 3.16 NLME-ED-SCAN.confirm.....	239
554	Table 3.17 NLME-JOIN.request	241
555	Table 3.18 NLME-JOIN.indication Parameters	243
556	Table 3.19 NLME-JOIN.confirm	244
557	Table 3.20 NLME-DIRECT-JOIN.request Parameters	245
558	Table 3.21 NLME-DIRECT-JOIN.confirm Parameters	246
559	Table 3.22 NLME-LEAVE.request Parameters	247
560	Table 3.23 NLME-LEAVE.indication Parameters	248
561	Table 3.24 NLME-LEAVE.confirm Parameters	249
562	Table 3.25 NLME-RESET.request Parameters	250
563	Table 3.26 NLME-RESET.confirm Parameters	251
564	Table 3.27 NLME-SYNC.request Parameters.....	252
565	Table 3.28 NLME-SYNC.confirm Parameters.....	254
566	Table 3.29 NLME-GET.request Parameters.....	256
567	Table 3.30 NLME-GET.confirm Parameters.....	257
568	Table 3.31 NLME-SET.request Parameters	258
569	Table 3.32 NLME-SET.confirm Parameters	259
570	Table 3.33 NLME-NWK-STATUS.indication Parameters	259
571	Table 3.34 NLME-ROUTE-DISCOVERY.request Parameters	260
572	Table 3.35 NLME_ROUTE-DISCOVERY.confirm Parameters	263
573	Table 3.36 Allowable Frame Control Sub-Field Configurations.....	264
574	Table 3.37 Values of the Frame Type Sub-Field.....	265
575	Table 3.38 Values of the Discover Route Sub-Field	265
576	Table 3.39 Values of the Multicast Mode Sub-Field	267
577	Table 3.40 NWK Command Frames	270
578	Table 3.41 Many-to-One Field Values	272
579	Table 3.42 Status Codes for Network Status Command Frame.....	277
580	Table 3.43 Fields of the End Device Timeout Request	290
581	Table 3.44 Requested Timeout Enumerated Values.....	290
582	Table 3.45 Payload fields of the End Device Timeout Response	292
583	Table 3.46 Enumeration of the End Device Timeout Response Status	292
584	Table 3.47 Values of the Parent Information Bitmask.....	293
585	Table 3.48 NWK Layer Constants.....	293
586	Table 3.49 NIB Attributes	294
587	Table 3.50 Route Record Table Entry Format.....	302
588	Table 3.51 Network Address Map.....	302
589	Table 3.52 Capability Information Bit-Fields.....	307
590	Table 3.53 Neighbor Table Entry Format.....	320
591	Table 3.54 Additional Neighbor Table Fields	323
592	Table 3.55 Example Addressing Offset Values for Each Given Depth within the Network	324
593	Table 3.56 Routing Table Entry	339
594	Table 3.57 Route Status Values.....	339
595	Table 3.58 Route Discovery Table Entry	340
596	Table 3.59 Broadcast Addresses.....	354
597	Table 3.60 Broadcast Transaction Record	356
598	Table 3.61 NWK Layer Information Fields.....	360
599	Table 3.62 NWK Layer Status Values.....	372
600	Table 4.1 Link Keys Used in ZigBee Networks	377
601	Table 4.2 NIB Security Attributes	383
602	Table 4.3 Elements of the Network Security Material Descriptor.....	384
603	Table 4.4 Elements of the Incoming Frame Counter Descriptor	385
604	Table 4.5 The APS Layer Security Primitives	386
605	Table 4.6 Security Policy for Accepting APS Commands in a Centralized Security Network	390
606	Table 4.7 Security Policy for Sending APS Commands in a Centralized Security Network	391
607	Table 4.8 APSME-TRANSPORT-KEY.request Parameters.....	393
608	Table 4.9 StandardKeyType Parameter of the Transport-Key, Verify-Key, and Confirm-Key Primitives.	393

609	Table 4.10 TransportKeyData Parameter for a Trust Center Link Key	394
610	Table 4.11 TransportKeyData Parameter for a Network Key.....	394
611	Table 4.12 TransportKeyData Parameter for an Application Link Key	394
612	Table 4.13 APSME-TRANSPORT-KEY.indication Parameters	397
613	Table 4.14 APSME-UPDATE-DEVICE.request Parameters.....	399
614	Table 4.15 APSME-UPDATE-DEVICE.indication Parameters	400
615	Table 4.16 APSME-REMOVE-DEVICE.request Parameters.....	401
616	Table 4.17 APSME-REMOVE-DEVICE.indication Parameters	402
617	Table 4.18 APSME-REQUEST-KEY.request Parameters	403
618	Table 4.19 RequestKeyType Values	404
619	Table 4.20 APSME-REQUEST-KEY.indication Parameters.....	405
620	Table 4.21 APSME-SWITCH-KEY.request Parameters.....	406
621	Table 4.22 APSME-SWITCH-KEY.indication Parameters	408
622	Table 4.23 APSME-VERIFY-KEY.request Parameters.....	410
623	Table 4.24 APSME-VERIFY-KEY.indication Parameters	411
624	Table 4.25 APSME-CONFIRM-KEY.request Parameters.....	413
625	Table 4.26 APSME-CONFIRM-KEY.indication Parameters	414
626	Table 4.27 Command Identifier Values.....	416
627	Table 4.29 AIB Security Attributes	423
628	Table 4.30 Elements of the Key-Pair Descriptor	423
629	Table 4.29 Security Levels Available to the NWK, and APS Layers.....	425
630	Table 4.30 Encoding of the Key Identifier Sub-Field.....	426
631	Table 4.31 Mapping of NLME-JOIN.indication Parameters to Update Device Status	431
632	Table 4.32 Trust Center Policy Values	443
633	Table 4.33 Joining Device Policy Values	452
634	Table D.1 Associate Request Header Fields.....	488
635	Table D.2 Data Request Header Fields.....	488
636	Table D.3 Association Response Header Fields	489
637	Table F.1 Internal Time-related Parameters	496
638	Table G.1 Semantics of the INTRP-DATA.request Primitive.....	501
639	Table G.2 Parameters of the GP-DATA.request primitive	503
640	Table G.3 Parameters of the INTRP-DATA.confirm	506
641	Table G.4 Parameters of the GP-DATA.confirm	507
642	Table G.5 Parameters of the GP-SEC.request	508
643	Table G.6 Parameters of the GP-SEC.response Primitive	510
644	Table G.7 Parameters of the INTRP-DATA.indication Primitive	512
645	Table G.8 Parameters of the GP-DATA.indication Primitive	516
646	Table G.9 MAC Header Fields for Inter-PAN APS Frames.....	520
647	Table G.10 Values for Frame Type for GPDF	522
648	Table G.11 Values of gpSecurityLevel.....	523

649 List of Figures

650	Figure 1.1 Outline of the ZigBee Stack Architecture	2
651	Figure 2.1 The APS Sub-Layer Reference Model	17
652	Figure 2.2 General APS Frame Format	42
653	Figure 2.3 Format of the Frame Control Field.....	42
654	Figure 2.4 Format of the Extended Header Sub-Frame	45
655	Figure 2.5 Format of the Extended Frame Control Field	45
656	Figure 2.6 Data Frame Format.....	46
657	Figure 2.7 APS Command Frame Format	47
658	Figure 2.8 Acknowledgement Frame Format	47
659	Figure 2.9. Binding on a Device Supporting a Binding Table.....	54
660	Figure 2.10 Successful Data Transmission Without an Acknowledgement	56
661	Figure 2.11 Successful Data Transmission with an Acknowledgement	57
662	Figure 2.12 Successful Data Transmission with Fragmentation.....	59
663	Figure 2.13 Fragmented Data Transmission with a Single Retransmission	60
664	Figure 2.14 Fragmented Data Transmission with Multiple Retransmissions	61
665	Figure 2.15 Format of the Complex Descriptor.....	66
666	Figure 2.16 Format of an Individual Complex Descriptor Field	66
667	Figure 2.17 Format of the MAC Capability Flags Field.....	69
668	Figure 2.18 Format of the Language and Character Set Field	75
669	Figure 2.19 Format of the ZDP Frame	82
670	Figure 2.20 Format of the NWK_addr_req Command	84
671	Figure 2.21 Format of the IEEE_addr_req_Command Frame.....	86
672	Figure 2.22 Format of the Node_Desc_req Command Frame	87
673	Figure 2.23 Format of the Power_Desc_req Command Frame	88
674	Figure 2.24 Format of the Simple_Desc_req Command Frame	88
675	Figure 2.25 Format of the Active_EP_req Command Frame	89
676	Figure 2.26 Format of the Match_Desc_req Command Frame	90
677	Figure 2.27 Format of the Complex_Desc_req Command Frame.....	91
678	Figure 2.28 Format of the User_Desc_req Command Frame	92
679	Figure 2.29 Format of the Discovery_Cache_req Command Frame	92
680	Figure 2.30 Format of the Device_ance Command Frame	93
681	Figure 2.31 Format of the Parent Annce Message.....	94
682	Figure 2.32 Format of the User_Desc_set Command Frame.....	96
683	Figure 2.33 Format of the System_Server_Discovery_req Command Frame	97
684	Figure 2.34 Format of the Discovery_Store_req Command Frame.....	97
685	Figure 2.35 Format of the Node_Desc_store_req Command Frame	98
686	Figure 2.36 Format of the Power_Desc_store_req Command Frame.....	99
687	Figure 2.37 Format of the Active_EP_store_req Command Frame	100
688	Figure 2.38 Format of the Simple_Desc_store_req Command Frame.....	101
689	Figure 2.39 Format of the Remove_node_cache_req Command Frame	102
690	Figure 2.40 Format of the Find_node_cache Command Frame	103
691	Figure 2.41 Format of the Extended_Simple_Desc_req Command Frame	104
692	Figure 2.42 Format of the Extended_Active_EP_req Command Frame	105
693	Figure 2.43 Format of the End_Device_Bind_req Command Frame	106
694	Figure 2.44 Format of the Bind_req Command Frame.....	108
695	Figure 2.45 Format of the Unbind_req Command Frame	109
696	Figure 2.46 Format of the Bind_Register_req Command Frame	111
697	Figure 2.47 Format of the Replace_Device_req Command Frame	111
698	Figure 2.48 Format of the Store_Bkup_Bind_Entry_req Command Frame	113
699	Figure 2.49 Format of the Remove_Bkup_Bind_Entry_req Command Frame	114
700	Figure 2.50 Format of the Backup_Bind_Table_req Command Frame.....	115
701	Figure 2.51 Fields of the Recover_Bind_Table_req Command Frame	116

702	Figure 2.52 Fields of the Backup_Source_Bind_req Command Frame	117
703	Figure 2.53 Format of the Recover_Source_Bind_req Command Frame	118
704	Figure 2.54 Format of the Mgmt_NWK_Disc_req Command Frame	119
705	Figure 2.55 Format of the Mgmt_Lqi_req Command Frame	120
706	Figure 2.56 Format of the Mgmt_Rtg_req Command Frame	121
707	Figure 2.57 Format of the Mgmt_Bind_req Command Frame	122
708	Figure 2.58 Format of the Mgmt_Leave_req Command Frame	122
709	Figure 2.59 Format of the Mgmt_Direct_Join_req Command Frame	124
710	Figure 2.60 Format of the Mgmt_Permit_Joining_req Command Frame	124
711	Figure 2.61 Fields of the Mgmt_Cache_req Command Frame.....	126
712	Figure 2.62 Fields of the Mgmt_NWK_Update_req Command Frame.....	126
713	Figure 2.63 Format of the NWK_addr_rsp Command Frame	130
714	Figure 2.64 Format of the IEEE_addr_rs Command Frame	132
715	Figure 2.65 Format of the Node_Desc_rsp Command Frame	133
716	Figure 2.66 Format of the Power_Desc_rsp Command Frame	135
717	Figure 2.67 Format of the Simple_Desc_rsp Command Frame	137
718	Figure 2.68 Format of the Active_EP_rsp Command Frame	138
719	Figure 2.69 Format of the Match_Desc_rsp Command Frame.....	139
720	Figure 2.70 Format of the Complex_Desc_rsp Command Frame	142
721	Figure 2.71 Format of the User_Desc_rsp Command Frame	144
722	Figure 2.72 System_Discovery_rsp Command Frame.....	145
723	Figure 2.73 Format of the User_Desc_conf Command Frame	146
724	Figure 2.74 Format of the Discovery_Cache_rsp Command Frame	147
725	Figure 2.75 Format of the Discovery_store_rsp Command Frame.....	148
726	Figure 2.76 Format of the Node_Desc_store_rsp Command Frame	149
727	Figure 2.77 Format of the Power_Desc_store_rsp Command Frame	150
728	Figure 2.78 Format of the Active_EP_store_rsp Command Frame	150
729	Figure 2.79 Format of the Simple_Desc_store_rsp Command Frame.....	151
730	Figure 2.80 Format of the Remove_node_cache_rsp Command Frame	152
731	Figure 2.81 Format of the Find_node_cache_rsp Command Frame.....	153
732	Figure 2.82 Format of the Extended_Simple_Desc_rsp Command Frame	154
733	Figure 2.83 Format of the Extended_Active_EP_rsp Command Frame	155
734	Figure 2.84 Format of the Parent_ance_rsp Command Frame	157
735	Figure 2.85 Format of the End_Device_Bind_rsp Command Frame	159
736	Figure 2.86 Format of the Bind_rsp Command Frame	160
737	Figure 2.87 Format of the Unbind_rsp Command Frame	161
738	Figure 2.88 Format of the Bind_Register_rsp Command Frame.....	162
739	Figure 2.89 Format of the Replace_Device_rsp Command Frame	163
740	Figure 2.90 Format of the Store_Bkup_Bind_Entry_rsp Command Frame	164
741	Figure 2.91 Format of the Remove_Bkup_Bind_Entry_rsp Command Frame	165
742	Figure 2.92 Format of the Backup_Bind_Table_rsp Command Frame	166
743	Figure 2.93 Format of the Backup_Bind_Table_rsp Command Frame	166
744	Figure 2.94 Format of the Backup_Source_Bind_rsp Command Frame	167
745	Figure 2.95 Format of the Recover_Source_Bind_rsp Command Frame	168
746	Figure 2.96 Format of the Mgmt_NWK_Disc_rsp Command Frame	170
747	Figure 2.97 Format of the Mgmt_Lqi_rsp Command Frame	172
748	Figure 2.98 Format of the Mgmt_Rtg_rsp Command Frame	175
749	Figure 2.99 Format of the Mgmt_Bind_rsp Command Frame	177
750	Figure 2.100 Format of the Mgmt_Leave_rsp Command Frame	179
751	Figure 2.101 Format of the Mgmt_Direct_Join_rsp Command Frame	179
752	Figure 2.102 Format of the Mgmt_Permit_Joining_rsp Command Frame	180
753	Figure 2.103 Format of the Mgmt_Cache_rsp Command Frame	181
754	Figure 2.104 Format of the Mgmt_NWK_Update_notify Command Frame	183
755	Figure 2.105 Primary Discovery Cache State Machine	187
756	Figure 2.106 Portability Message Sequence Chart: ZED Secured Rejoin.....	197
757	Figure 2.107 Portability Message Sequence Chart: ZR/ZED Trust Center Rejoin.....	198

758	Figure 3.1 The NWK Layer Reference Model	220
759	Figure 3.2 Message Sequence Chart for Resetting the Network Layer	252
760	Figure 3.3 Message Sequence Chart for Synchronizing in a Non-Beaconing Network	255
761	Figure 3.4 Message Sequence Chart for Synchronizing in a Beacon-Enabled Network	255
762	Figure 3.5 General NWK Frame Format	264
763	Figure 3.6 Frame Control Field	264
764	Figure 3.7 Multicast Control Field Format.....	267
765	Figure 3.8 Source Route Subframe Format	268
766	Figure 3.9 Data Frame Format.....	268
767	Figure 3.10 NWK Command Frame Format	269
768	Figure 3.11 Route Request Command Frame Format	271
769	Figure 3.12 Route Request Command Options Field	271
770	Figure 3.13 Route Reply Command Format.....	274
771	Figure 3.14 Route Reply Command Options Field	275
772	Figure 3.15 Network Status Command Frame Format.....	276
773	Figure 3.16 Leave Command Frame Format.....	279
774	Figure 3.17 Leave Command Options Field.....	280
775	Figure 3.18 Route Record Command Format.....	280
776	Figure 3.19 Rejoin Request Command Frame Format	281
777	Figure 3.20 Rejoin Response Command Frame Format.....	282
778	Figure 3.21 Link Status Command Format	284
779	Figure 3.22 Link Status Command Options Field	284
780	Figure 3.23 Link Status Entry.....	285
781	Figure 3.24 Network Report Command Frame Format.....	285
782	Figure 3.25 Network Report Command Options Field.....	286
783	Figure 3.26 Report Command Identifier Sub-Field	287
784	Figure 3.27 PAN Identifier Conflict Report	287
785	Figure 3.28 Network Update Command Frame Format	287
786	Figure 3.29 Network Update Command Options Field	288
787	Figure 3.30 Update Command Identifier Sub-Field	289
788	Figure 3.31 PAN Identifier Update.....	289
789	Figure 3.32 Format of the End Device Timeout Request Command.....	289
790	Figure 3.33 Format of the End Device Timeout Response Command	291
791	Figure 3.34 Establishing a New Network	304
792	Figure 3.35 Permitting Devices to Join a Network.....	305
793	Figure 3.36 Procedure for Joining a Network Through Association	310
794	Figure 3.37 Procedure for Handling a Join Request	312
795	Figure 3.38 Child Rejoin Procedure	314
796	Figure 3.39 Parent Rejoin Procedure.....	316
797	Figure 3.40 Joining a Device to a Network Directly	317
798	Figure 3.41 Child Procedure for Joining or Re-Joining a Network through Orphaning.....	318
799	Figure 3.42 Parent Procedure for Joining or Re-Joining a Device to Its Network through Orphaning	319
800	Figure 3.43 Address Assignment in an Example Network	325
801	Figure 3.44 Initiation of the Leave Procedure	328
802	Figure 3.45 Procedure for a Device to Remove Its Child.....	329
803	Figure 3.46 On Receipt of a Leave Command	330
804	Figure 3.47 On Receipt of a Leave Command by a ZED	331
805	Figure 3.48 Typical Frame Structure for a Beaconing Device	352
806	Figure 3.49 Parent-Child Superframe Positioning Relationship.....	354
807	Figure 3.50 Broadcast Transaction Message Sequence Chart	357
808	Figure 3.51 Format of the MAC Sub-Layer Beacon Payload.....	362
809	Figure 3.52 Initial Setup of the End Device Timeout.....	366
810	Figure 3.53 Child Keepalive: MAC Data Poll Method	366
811	Figure 3.54 Child Keepalive: End Device Timeout Request Method.....	367
812	Figure 3.55 Aging out Children: MAC Data Poll Method - Secure Rejoin.....	368
813	Figure 3.56 Aging out Children: MAC Data Poll - Trust Center Rejoin.....	369

814	Figure 3.57 Aging out Children: End Device Timeout Request Method - Secure Rejoin.....	370
815	Figure 3.58 Aging out Children: End Device Timeout Request Method - Trust Center Rejoin.....	371
816	Figure 4.1 ZigBee Frame with Security on the NWK Level	378
817	Figure 4.2 ZigBee Frame with Security on the APS Level	379
818	Figure 4.3 Secured NWK Layer Frame Format.....	383
819	Figure 4.4 Request Key Service Processing for Trust Center Link Key.....	403
820	Figure 4.5 Verify-Key Processing	409
821	Figure 4.6 Secured APS Layer Frame Format.....	415
822	Figure 4.7 Transport-Key Command Frame.....	417
823	Figure 4.8 Trust Center Link Key Descriptor Field in Transport-Key Command.....	417
824	Figure 4.9 Network Key Descriptor Field in Transport-Key Command	418
825	Figure 4.10 Application Link Key Descriptor in Transport-Key Command	418
826	Figure 4.11 Update-Device Command Frame Format.....	418
827	Figure 4.12 Remove-Device Command Frame Format.....	419
828	Figure 4.13 Request-Key Command Frame Format.....	419
829	Figure 4.14 Switch-key Command Frame Format	420
830	Figure 4.15 Tunnel Command Frame Format	420
831	Figure 4.16 Verify-Key Command Frame.....	421
832	Figure 4.17 Confirm-Key Command Frame.....	422
833	Figure 4.18 Auxiliary Frame Header Format	424
834	Figure 4.19 Security Control Field Format.....	425
835	Figure 4.20 CCM Nonce	427
836	Figure 4.21 Example of Joining a Secured Network	429
837	Figure 4.22 - Multi-hop Join and Trust Center Rejoin Diagram.....	433
838	Figure 4.23 - Secure Rejoin	434
839	Figure 4.24 - Trust Center Rejoin.....	435
840	Figure 4.25 Example Network Key-Update Procedure	437
841	Figure 4.26 Example End-to-End Application Key Establishment Procedure	439
842	Figure 4.27 Example Remove-Device Procedure.....	440
843	Figure 4.28 Example Device-Leave Procedure	441
844	Figure B.1 Symmetric-Key Authenticated Key Agreement Scheme.....	465
845	Figure B.2 Mutual Symmetric-Key Entity Authentication Scheme	468
846	Figure G.1 ZigBee Stack with Inter-PAN APS	500
847	Figure G.2 - ZigBee Frame Format Overview.....	519
848	Figure G.3 Inter-PAN Frame Format	519
849	Figure G.4 Green Power Device Frame Format	519
850	Figure G.5 Stub NWK Header for Inter-PAN messages	521
851	Figure G.6 NWK Header Frame Control for Green Power Device Frames	521
852	Figure G.7 NWK Header Frame Control for Green Power Device Frames	521
853	Figure G.8 Format of Extended NWK Frame Control field for GPDF with Application ID 0b000 and 0b010522	
854	Figure G.9 Inter-PAN APS Header Format.....	524
855	Figure G.10 Format of the APS Frame Control Field for Inter-PAN Messages.....	524
856	Figure G.11 Format of the AES Nonce for Green Power Device Frames	528
857	Figure G.12 Format of the Security Control field of the AES Nonce for Green Power Device Frames	529
858		
859		
860		
861		
862		
863		
864		
865		

866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885

This page intentionally left blank.

886

CHAPTER 1 ZIGBEE PROTOCOL OVERVIEW

887

888

1.1 Protocol Description

889
890
891
892

The ZigBee Alliance has developed a very low-cost, very low-power-consumption, two-way, wireless communications standard. Solutions adopting the ZigBee standard will be embedded in consumer electronics, home and building automation, industrial controls, PC peripherals, medical sensor applications, toys, and games.

893

1.1.1 Scope

894
895
896
897

This document contains specifications, interface descriptions, object descriptions, protocols and algorithms pertaining to the ZigBee protocol standard, including the application support sub-layer (APS), the ZigBee device objects (ZDO), ZigBee device profile (ZDP), the application framework, the network layer (NWK), and ZigBee security services.

898

1.1.2 Purpose

899
900
901
902

The purpose of this document is to provide a definitive description of the ZigBee protocol standard as a basis for future implementations, such that any number of companies incorporating the ZigBee standard into platforms and devices on the basis of this document will produce interoperable, low-cost, and highly usable products for the burgeoning wireless marketplace.

903

1.1.3 Stack Architecture

904
905
906
907

The ZigBee stack architecture is made up of a set of blocks called layers. Each layer performs a specific set of services for the layer above. A data entity provides a data transmission service and a management entity provides all other services. Each service entity exposes an interface to the upper layer through a service access point (SAP), and each SAP supports a number of service primitives to achieve the required functionality.

908
909
910
911
912

The IEEE 802.15.4 standard defines the two lower layers: the physical (PHY) layer and the medium access control (MAC) sub-layer. The ZigBee Alliance builds on this foundation by providing the network (NWK) layer and the framework for the application layer. The application layer framework consists of the application support sub-layer (APS) and the ZigBee device objects (ZDO). Manufacturer-defined application objects use the framework and share APS and security services with the ZDO.

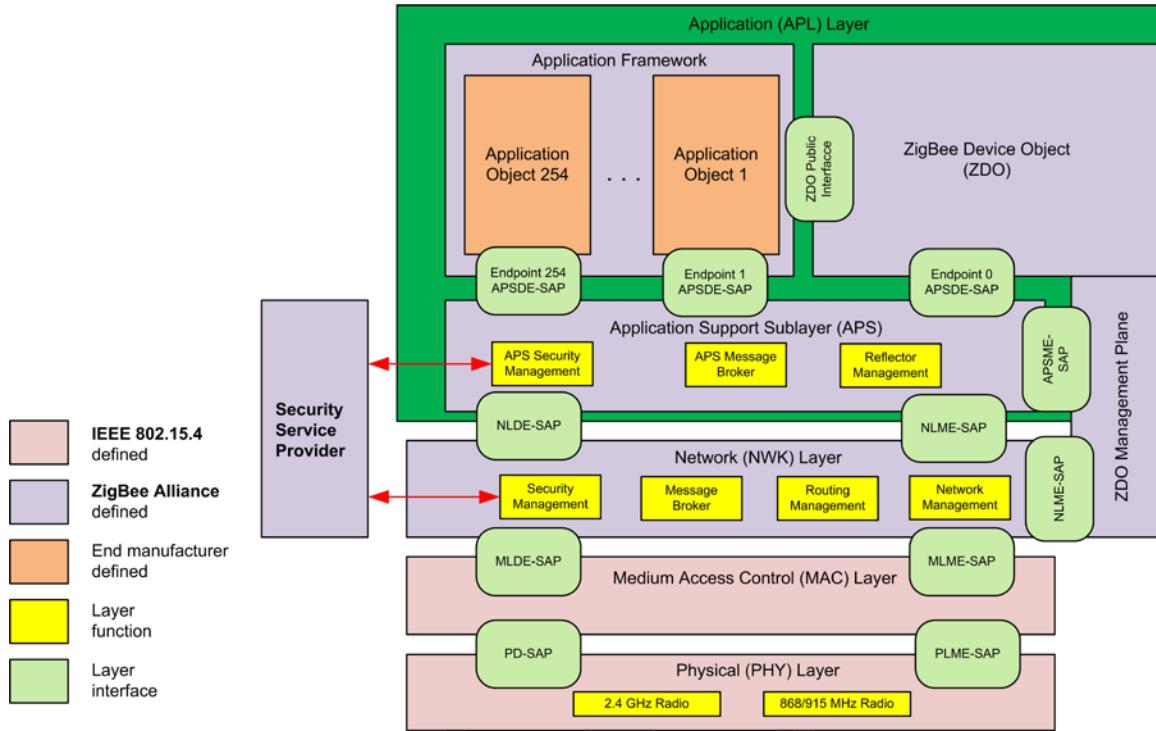
913
914
915
916

The PHY layer operates in two separate frequency ranges: 868/915 MHz and 2.4 GHz. The lower frequency PHY layer covers both the 868 MHz European band and the 915 MHz band, used in countries such as the United States and Australia. The higher frequency PHY layer is used virtually worldwide. A complete description of the PHY layers can be found in [B1].

917 The MAC sub-layer controls access to the radio channel using a CSMA-CA mechanism. Its responsibilities
918 may also include transmitting beacon frames, synchronization, and providing a reliable transmission
919 mechanism. A complete description of the IEEE 802.15.4 MAC sub-layer can be found in [B1]. Figure 1.1
920 represents the outline of the ZigBee Stack Architecture.

921

Figure 1.1 Outline of the ZigBee Stack Architecture



922

1.1.4 Network Topology

924 The ZigBee network layer (NWK) supports star, tree, and mesh topologies. In a star topology, the network is
925 controlled by one single device called the ZigBee coordinator. The ZigBee coordinator is responsible for
926 initiating and maintaining the devices on the network. All other devices, known as end devices, directly
927 communicate with the ZigBee coordinator. In mesh and tree topologies, the ZigBee coordinator is responsi-
928 ble for starting the network and for choosing certain key network parameters, but the network may be ex-
929 tended through the use of ZigBee routers. In tree networks, routers move data and control messages through
930 the network using a hierarchical routing strategy. Tree networks may employ beacon-oriented communica-
931 tion as described in the IEEE 802.15.4 specification. Mesh networks allow full peer-to-peer communication.
932 ZigBee routers in mesh networks do not currently emit regular IEEE 802.15.4 beacons. This specification
933 describes only intra-PAN networks, that is, networks in which communications begin and terminate within
934 the same network.

935

1.2 Conventions and Abbreviations

1.2.1 Symbols and Notation

936 Notation follows from ANSI X9.63-2001, §2.2 [B7].

937

938 **1.2.2 Integers, Octets, and Their Representation**

939 Throughout Annexes A through D, the representation of integers as octet strings and of octet strings as binary
940 strings shall be fixed. All integers shall be represented as octet strings in most-significant-octet first order.
941 This representation conforms to the convention in Section 4.3 of [B7]. All octets shall be represented as
942 binary strings in most-significant-bit first order.

943 **1.2.3 Transmission Order**

944 Unless otherwise indicated, the transmission order of all frames in this specification follows the conventions
945 used in [B1]:

- 946 • Frame formats are depicted in the order in which they are transmitted by the PHY layer—from left to
947 right—where the leftmost bit is transmitted first in time.
948 • Bits within each field are numbered from 0 (leftmost, and least significant) to k-1 (rightmost, and most
949 significant), where the length of the field is k bits.
950 • Fields that are longer than a single octet are sent to the PHY in order from the octet containing the lowest
951 numbered bits to the octet containing the highest-numbered bits.

952 **1.2.4 Strings and String Operations**

953 A string is a sequence of symbols over a specific set (for example, the binary alphabet {0,1} or the set of all
954 octets). The length of a string is the number of symbols it contains (over the same alphabet). The empty
955 string has length 0. The right-concatenation of two strings x and y of length m and n respectively (notation:
956 $x // y$), is the string z of length $m+n$ that coincides with x on its leftmost m symbols and with y on its right-
957 most n symbols. An octet is a symbol string of length 8. In our context, all octets are strings over the binary
958 alphabet.

959 **1.3 Acronyms and Abbreviations**

960 For the purposes of this standard, the following acronyms and abbreviations apply:

Acronym or Abbreviation	Definition
AIB	Application support layer information base
AF	Application framework
APDU	Application support sub-layer protocol data unit
APL	Application layer
APS	Application support sub-layer
APSDE	Application support sub-layer data entity
APSDE-SAP	Application support sub-layer data entity – service access point
APSME	Application support sub-layer management entity

Acronym or Abbreviation	Definition
APSME-SAP	Application support sub-layer management entity – service access point
ASDU	APS service data unit
BRT	Broadcast retry timer
BTR	Broadcast transaction record
BTT	Broadcast transaction table
CCM*	Carrier sense multiple access – collision avoidance
CSMA-CA	Carrier sense multiple access – collision avoidance.
EPID	Extended PAN ID
FFD	Full function device
GPD	Green Power Device
GPDF	Green Power Device Frame
GPEP	Green Power Endpoint
GTS	Guaranteed time slot
HDR	Header
IB	Information base
LQI	Link quality indicator
LR-WPAN	Low rate wireless personal area network
MAC	Medium access control
MCPS-SAP	Medium access control common part sub-layer service access point
MIC	Message integrity code
MLME-SAP	Medium access control sub-layer management entity service access point
MSC	Message sequence chart

Acronym or Abbreviation	Definition
MSDU	Medium access control sub-layer service data unit
MSG	Message service type
NBDT	Network broadcast delivery time
NHLE	Next higher layer entity
NIB	Network layer information base
NLDE	Network layer data entity
NLDE-SAP	Network layer data entity – service access point
NLME	Network layer management entity
NLME-SAP	Network layer management entity – service access point
NPDU	Network layer protocol data unit
NSDU	Network service data unit
NWK	Network
OSI	Open systems interconnection
PAN	Personal area network
PD-SAP	Physical layer data service access point
PDU	Protocol data unit
PHY	Physical layer
PIB	Personal area network information base
PLME-SAP	Physical layer management entity – service access point
POS	Personal operating space
QOS	Quality of service
RFD	Reduced function device

Acronym or Abbreviation	Definition
RREP	Route reply
RREQ	Route request
RN	Routing node
SAP	Service access point
SKG	Secret key generation
SSP	Security services provider
SSS	Security services specification
WPAN	Wireless personal area network
XML	Extensible markup language
ZB	ZigBee
ZDO	ZigBee device object

961 **1.4 Glossary**

962 **1.4.1 Definitions**

963 **1.4.1.1 Conformance Levels**

964 The conformance level definitions shall follow those in clause 13, section 1 of [B14].

965 **Expected:** A key word used to describe the behavior of the hardware or software in the design models
966 assumed by this Specification. Other hardware and software design models may also be implemented.

967 **May:** A key word indicating a course of action permissible within the limits of the standard (may
968 equals is permitted to).

969 **Shall:** A key word indicating mandatory requirements to be strictly followed in order to conform to the
970 standard; deviations from shall are prohibited (*shall equals is required to*).

971 **Should:** A key word indicating that, among several possibilities, one is recommended as being partic-
972 ularly suitable, without mentioning or excluding others; that a certain course of action is preferred but
973 not necessarily required; or, that (in the negative form) a certain course of action is deprecated but not
974 prohibited (*should equals is recommended that*).

975 **Reserved Codes:** A set of codes that are defined in this specification, but not otherwise used. Future
976 specifications may implement the use of these codes. A product implementing this specification shall
977 not generate these codes.

978 **Reserved Fields:** A set of fields that are defined in this specification, but are not otherwise used.
979 Products that implement this specification shall zero these fields and shall make no further assumptions
980 about these fields nor perform processing based on their content.

981 **ZigBee Protocol Version:** The name of the ZigBee protocol version governed by this specification.
982 The protocol version sub-field of the frame control field in the NWK header of all ZigBee Protocol
983 Stack frames conforming to this specification shall have a value of 0x02 for all ZigBee frames, and a
984 value of 0x03 for the ZigBee Green Power frames. The protocol version support required by various
985 ZigBee specification revisions appears in Table 1.1.

986 **Table 1.1 ZigBee Protocol Versions**

Specification	Protocol Version	Comment
ZigBee Green Power	0x03	ZigBee Green Power feature. See Annex G.
ZigBee Pro ZigBee 2006	0x02	Backwards compatibility not required. ZigBee Pro and ZigBee 2006 compatibility required.
ZigBee 2004	0x01	Original ZigBee version.

987 A ZigBee device that conforms to this version of the specification may elect to provide backward compatibility
988 with the 2004 revision of the specification. If it so elects, it shall do so by supporting, in addition to
989 the frame formats and features described in this specification version, all frame formats and features as
990 specified in the older version. [All devices in an operating network, regardless of which revisions of the
991 ZigBee specification they support internally, shall, with respect to their external, observable behavior, con-
992 sistently conform to a single ZigBee protocol version.] A single ZigBee network shall not contain devices
993 that conform, in terms of their external behavior, to multiple ZigBee protocol versions. [The protocol ver-
994 sion of the network to join shall be determined by a backwardly compatible device in examining the beacon
995 payload prior to deciding to join the network; or shall be established by the application if the device is a
996 ZigBee coordinator.] A ZigBee device conforming to this specification may elect to support only protocol
997 version 0x02, whereby it shall join only networks that advertise commensurate beacon payload support. A
998 ZigBee device that conforms to this specification shall discard all frames carrying a protocol version
999 sub-field value other than 0x01, 0x02, or 0x03. It shall process only protocol versions of 0x01 or 0x02,
1000 consistent with the protocol version of the network that the device participates within. A ZigBee device that
1001 conforms to this specification shall pass the frames carrying the protocol version sub-field value 0x03 to
1002 the Interpan APS (see Annex G), if it supports the ZigBee Green Power, otherwise it shall drop them.

1003 **1.4.1.2 ZigBee Definitions**

1004 For the purposes of this standard, the following terms and definitions apply. Terms not defined in this sec-
1005 tion can be found in [B1] or in [B7].

1006 **Access control list:** This is a table used by a device to determine which devices are authorized to per-
1007 form a specific function. This table may also store the security material (for example, cryptographic
1008 keys, frame counts, key counts, security level information) used for securely communicating with other
1009 devices.

1010 **Active network key:** This is the key used by a ZigBee device to secure outgoing NWK frames and
1011 that is available for use to process incoming NWK frames.

1012 **Alternate network key:** This is a key available to process incoming NWK frames in lieu of the active
1013 network key.

1014 **Application domain:** This describes a broad area of applications, such as building automation.

1015 **Application key:** This is a link key transported by the Trust center to a device for the purpose of se-
1016 curing end-to-end communication.

- 1017 **Application object:** This is a component of the top portion of the application layer defined by the
1018 manufacturer that actually implements the application.
- 1019 **Application profile:** This is a collection of device descriptions, which together form a cooperative ap-
1020 plication. For instance, a thermostat on one node communicates with a furnace on another node. To-
1021 gether, they cooperatively form a heating application profile.
- 1022 **Application support sub-layer protocol data unit:** This is a unit of data that is exchanged between
1023 the application support sub-layers of two peer entities.
- 1024 **APS command frame:** This is a command frame from the APSME on a device addressed to the peer
1025 entity on another device.
- 1026 **Association:** This is the service provided by the IEEE 802.15.4 MAC sub-layer that is used to estab-
1027 lish membership in a network.
- 1028 **Attribute:** This is a data entity which represents a physical quantity or state. This data is communi-
1029 cated to other devices using commands.
- 1030 **Beacon-enabled personal area network:** This is a personal area network containing at least one de-
1031 vice that transmits beacon frames at a regular interval.
- 1032 **Binding:** This is the creation of a unidirectional logical link between a source endpoint/cluster identi-
1033 fier pair and a destination endpoint, which may exist on one or more devices.
- 1034 **Broadcast:** This is the transmission of a message to every device in a particular PAN belonging to one
1035 of a small number of statically defined broadcast groups, for example all routers, and within a given
1036 transmission radius measured in hops.
- 1037 **Broadcast jitter:** This is a random delay time introduced by a device before relaying a broadcast
1038 transaction.
- 1039 **Broadcast transaction record:** This is a local receipt of a broadcast message that was either initiated
1040 or relayed by a device.
- 1041 **Broadcast transaction table:** This is a collection of broadcast transaction records.
- 1042 **Cluster:** This is an application message, which may be a container for one or more attributes. As an
1043 example, the ZigBee Device Profile defines commands and responses. These are contained in Clusters
1044 with the cluster identifiers enumerated for each command and response. Each ZigBee Device Profile
1045 message is then defined as a cluster. Alternatively, an application profile may create sub-types within
1046 the cluster known as attributes. In this case, the cluster is a collection of attributes specified to accom-
1047 pany a specific cluster identifier (sub-type messages.)
- 1048 **Cluster identifier:** This is a reference to an enumeration of clusters within a specific application pro-
1049 file or collection of application profiles. The cluster identifier is a 16-bit number unique within the
1050 scope of each application profile and identifies a specific cluster. Conventions may be established
1051 across application profiles for common definitions of cluster identifiers whereby each application pro-
1052 file defines a set of cluster identifiers identically. Cluster identifiers are designated as inputs or outputs
1053 in the simple descriptor for use in creating a binding table.
- 1054 **Coordinator:** This is an IEEE 802.15.4 device responsible for associating and disassociating devices
1055 into its PAN. A coordinator must be a full-function device (FFD).
- 1056 **Data integrity:** This is assurance that the data has not been modified from its original form.
- 1057 **Data key:** This is a key derived from a link key used to protect data messages.
- 1058 **Device:** This is any entity that contains an implementation of the ZigBee protocol stack.
- 1059 **Device application:** This is a special application that is responsible for Device operation. The device
1060 application resides on endpoint 0 by convention and contains logic to manage the device's networking
1061 and general maintenance features. Endpoints 241-254 are reserved for use by the Device application or
1062 common application function agreed within the ZigBee Alliance.

1063 **Device description:** This is a description of a specific device within an application profile. For example, the light sensor device description is a member of the home automation application profile. The device description also has a unique identifier that is exchanged as part of the discovery process.

1066 **Direct addressing:** This is a mode of addressing in which the destination of a frame is completely specified in the frame itself.

1068 **Direct transmission:** This is a frame transmission using direct addressing.

1069 **Disassociation:** This is the service provided by the IEEE 802.15.4 MAC sub-layer that is used to discontinue the membership of a device in a network.

1071 **End application:** This is for applications that reside on endpoints 1 through 254 on a Device. The end applications implement features that are non-networking and ZigBee protocol related. Endpoints 241 through 254 shall only be used by the End application with approval from the ZigBee Alliance. The Green Power cluster, if implemented, SHALL use endpoint 242.

1075 **End device binding:** This is the procedure for creating or removing a binding link initiated by each of the end devices that will form the link. The procedure may or may not involve user intervention.

1077 **Endpoint:** This is a particular component within a unit. Each ZigBee device may support up to 254 such components.

1079 **Endpoint address:** This is the address assigned to an endpoint. This address is assigned in addition to the unique, 64-bit IEEE address and 16-bit network address.

1081 **Extended PAN ID:** This is the globally unique 64-bit PAN identifier of the network. This identifier should be unique among the PAN overlapping in a given area. This identifier is used to avoid PAN ID conflicts between distinct networks.

1084 **Information base:** This is a collection of variables that define certain behavior in a layer. These variables can be specified or obtained from a layer through its management service.

1086 **Key establishment:** This is a mechanism that involves the execution of a protocol by two devices to derive a mutually shared secret key.

1088 **Key-load key:** This is a key derived from a link key used to protect key transport messages carrying a link key.

1090 **Key transport:** This is a mechanism for communicating a key from one device to another device or other devices.

1092 **Key-transport key:** This is a key derived from a link key used to protect key transport messages carrying a key.

1094 **Key update:** This is a mechanism implementing the replacement of a key shared amongst two or more devices by means of another key available to that same group.

1096 **Local device:** This is the initiator of a ZDP command.

1097 **Link key:** This is a key that is shared exclusively between two, and only two, peer application-layer entities within a PAN.

1099 **Mesh network:** This is a network in which the routing of messages is performed as a decentralized, cooperative process involving many peer devices routing on each other's behalf.

1101 **Multicast:** This is a transmission to every device in a particular PAN belonging to a dynamically defined multicast group, and within a given transmission radius measured in hops.

1103 **Multihop network:** This is a network, in particular a wireless network, in which there is no guarantee that the transmitter and the receiver of a given message are connected or linked to each other. This implies that intermediate devices must be used as routers.

1106 **Non-beacon-enabled personal area network:** This is a personal area network that does not contain any devices that transmit beacon frames at a regular interval.

- 1108 **Neighbor table:** This is a table used by a ZigBee device to keep track of other devices within the POS.
- 1109 **Network address:** This is the address assigned to a device by the network layer and used by the net-
- 1110 work layer for routing messages between devices.
- 1111 **Network broadcast delivery time:** This is the time required by a broadcast transaction to reach every
- 1112 device of a given network.
- 1113 **Network manager:** This is a ZigBee device that implements network management functions as de-
- 1114 scribed in Clause 3, including PAN ID conflict resolution and frequency agility measures in the face of
- 1115 interference.
- 1116 **Network protocol data unit:** This is a unit of data that is exchanged between the network layers of
- 1117 two peer entities.
- 1118 **Network service data unit:** This is the information that is delivered as a unit through a network ser-
- 1119 vice access point.
- 1120 **Node:** This is a collection of independent device descriptions and applications residing in a single unit
- 1121 and sharing a common 802.15.4 radio.
- 1122 **Normal operating state:** This is the processing which occurs after all startup and initialization pro-
- 1123 cessing has occurred and prior to initiation of shutdown processing.
- 1124 **NULL:** a parameter or variable value that means unspecified, undefined, or unknown. The exact value
- 1125 of NULL is implementation-specific, and must not conflict with any other parameters or values.
- 1126 **Octet:** eight bits of data, used as a synonym for a byte.
- 1127 **OctetDuration:** transmission time (in seconds) of an octet on PHY layer. This time is calculated as
- 1128 $8/\text{phyBitRate}$ where phyBitRate can be found in Table 1 of [B1]. To get milliseconds from N Oc-
- 1129 tetDurations for 2.4 GHz the follow formula has to be used: $N*(8/250000)*1000$ where 250000 bit rate
- 1130 on 2.4 GHz and 8 number of bits in one octet.
- 1131 **One-way function:** a function whose forward computation is much easier to perform than its inverse.
- 1132 **Orphaned device:** a device, typically a ZigBee end device that has lost communication with the
- 1133 ZigBee device through which it has its PAN membership.
- 1134 **PAN coordinator:** the principal controller of an IEEE 802.15.4-based network that is responsible for
- 1135 network formation. The PAN coordinator must be a full function device (FFD).
- 1136 **PAN information base:** a collection of variables in the IEEE 802.15.4 standard that are passed be-
- 1137 tween layers, in order to exchange information. This database may include the access control list,
- 1138 which stores the security material.
- 1139 **Personal operating space:** the area within reception range of a single device.
- 1140 **Private method:** attributes and commands which are accessible to ZigBee device objects only and
- 1141 unavailable to the end applications.
- 1142 **Protocol data unit:** the unit of data that is exchanged between two peer entities.
- 1143 **Public method:** attributes and commands which are accessible to end applications.
- 1144 **Radio:** the IEEE 802.15.4 radio that is part of every ZigBee device.
- 1145 **Remote device:** the target of a ZDP command.
- 1146 **Route discovery:** an operation in which a ZigBee coordinator or ZigBee router attempts to discover a
- 1147 route to a remote device by issuing a route request command frame.
- 1148 **Route discovery table:** a table used by a ZigBee coordinator or ZigBee router to store temporary in-
- 1149 formation used during route discovery.
- 1150 **Route reply:** a ZigBee network layer command frame used to reply to route requests.

1151 **Route request:** a ZigBee network layer command frame used to discover paths through the network
1152 over which subsequent messages may be delivered.

1153 **Routing table:** a table in which a ZigBee coordinator or ZigBee router stores information required to
1154 participate in the routing of frames.

1155 **Service discovery:** the ability of a device to locate services of interest.

1156 **Stack profile:** an agreement by convention outside the scope of the ZigBee specification on a set of
1157 additional restrictions with respect to features declared optional by the specification itself.

1159 **Trust center:** the device trusted by devices within a ZigBee network to distribute keys for the purpose
1160 of network and end-to-end application configuration management.

1161 **Unicast:** the transmission of a message to a single device in a network.

1162 **ZigBee coordinator:** an IEEE 802.15.4 PAN coordinator.

1163 **ZigBee device object:** the portion of the application layer responsible for defining the role of the de-
1164 vice within the network (for example, ZigBee coordinator or end device), initiating and/or responding
1165 to binding and discovery requests, and establishing a secure relationship between network devices.

1166 **ZigBee end device:** an IEEE 802.15.4 RFD or FFD participating in a ZigBee network, which is nei-
1167 ther the ZigBee coordinator nor a ZigBee router.

1168 **ZigBee router:** an IEEE 802.15.4 FFD participating in a ZigBee network, which is not the ZigBee co-
1169 ordinator but may act as an IEEE 802.15.4 coordinator within its personal operating space, that is ca-
1170 pable of routing messages between devices and supporting associations.

1171 1.5 References

1172 The following standards contain provisions, which, through reference in this document, constitute provi-
1173 sions of this standard. Normative references are given in ZigBee/IEEE References and Normative Refer-
1174 ences and informative references are given in Informative References At the time of publication, the edi-
1175 tions indicated were valid. All standards are subject to revision, and parties to agreements based on this
1176 standard are encouraged to investigate the possibility of applying the most recent editions of the references,
1177 as indicated in this section.

1178 1.5.1 ZigBee/IEEE References

- 1179 [B1] 802.15.4-2011, IEEE Standard for Local and metropolitan area networks--Part 15.4: Low-Rate Wireless
1180 Personal Area Networks (LR-WPANs)
- 1181 [B3] Document 03-285r00: Suggestions for the Improvement of the IEEE 802.15.4 Standard, July 2003.
- 1182 [B4] Document 09-5499r26: Green Power specification

1183 1.5.2 Normative References

- 1184 [B5] ISO/IEC 639-1:2002 Codes for the representation of names of languages — Part 1: Alpha-2 code.
- 1185 [B6] ISO/IEC 646:199 Information technology — ISO 7-bit coded character set for information interchange.
- 1186 [B7] ANSI X9.63-2001, Public Key Cryptography for the Financial Services Industry - Key Agreement and Key
1187 Transport Using Elliptic Curve Cryptography, American Bankers Association, November 20, 2001. Avail-
1188 able from <http://www.ansi.org>.

- 1189 [B8] FIPS Pub 197, Advanced Encryption Standard (AES), Federal Information Processing Standards Publication 197, US Department of Commerce/N.I.S.T., Springfield, Virginia, November 26, 2001. Available from <http://csrc.nist.gov/>.
- 1192 [B9] FIPS Pub 198, The Keyed-Hash Message Authentication Code (HMAC), Federal Information Processing Standards Publication 198, US Department of Commerce/N.I.S.T., Springfield, Virginia, March 6, 2002. Available from <http://csrc.nist.gov/>.
- 1195 [B10] ISO/IEC 9798-2, Information Technology - Security Techniques — Entity Authentication Mechanisms — Part 2: Mechanisms Using Symmetric Encipherment Algorithms, International Standardization Organization, Geneva, Switzerland, 1994 (first edition). Available from <http://www.iso.org/>.
- 1198 [B11] NIST Pub 800-38A 2001 ED, Recommendation for Block Cipher Modes of Operation — Methods and Techniques, NIST Special Publication 800-38A, 2001 Edition, US Department of Commerce/N.I.S.T., December 2001. Available from <http://csrc.nist.gov/>.
- 1201 [B12] NIST, Random Number Generation and Testing. Available from <http://csrc.nist.gov/rng/>.

1202 1.5.3 Informative References

- 1203 [B13] FIPS Pub 140-2, Security requirements for Cryptographic Modules, US Department of Commerce/N.I.S.T., Springfield, Virginia, June 2001 (supersedes FIPS Pub 140-1). Available from <http://csrc.nist.gov/>.
- 1205 [B14] IEEE Standards Style Manual, published and distributed in May 2000 and revised on September 20, 2001. Available from <http://standards.ieee.org/guides/style/>.
- 1207 [B15] ISO/IEC 7498-1:1994 Information technology — Open systems interconnection — Basic reference model: The basic model.
- 1209 [B16] ISO/IEC 10731:1994, Information technology — Open Systems Interconnection — Conventions for the definition of OSI services.
- 1211 [B17] ISO/IEC 9646-1:1991, Information technology — Open systems Interconnection — Conformance testing methodology and framework — Part 1: General concepts.
- 1213 [B18] ISO/IEC 9646-7:1995, Information technology — Open Systems Interconnection — Conformance testing methodology and framework — Part 7. Implementation conformance statements.
- 1215 [B19] A.J. Menezes, P.C. van Oorschot, S.A. Vanstone, Handbook of Applied Cryptography, Boca Raton: CRC Press, 1997.
- 1217 [B20] FIPS Pub 113, Computer Data Authentication, Federal Information Processing Standard Publication 113, US Department of Commerce/N.I.S.T., May 30, 1985. Available from <http://csrc.nist.gov/>.
- 1219 [B21] R. Housley, D. Whiting, N. Ferguson, Counter with CBC-MAC (CCM), submitted to N.I.S.T., June 3, 2002. Available from <http://csrc.nist.gov/encryption/modules/proposedmodes/>.
- 1221 [B22] J. Jonsson, On the Security of CTR + CBC-MAC, in Proceedings of Selected Areas in Cryptography — SAC 2002, K. Nyberg, H. Heys, Eds., Lecture Notes in Computer Science, Vol. 2595, pp. 76-93, Berlin: Springer, 2002.
- 1224 [B23] J. Jonsson, On the Security of CTR + CBC-MAC, NIST Mode of Operation — Additional CCM Documentation. Available from <http://csrc.nist.gov/encryption/modes/proposedmodes/>.
- 1226 [B24] P. Rogaway, D. Wagner, A Critique of CCM, IACR ePrint Archive 2003-070, April 13, 2003.
- 1227 [B25] ZigBee Document 053298- CSG Framework Profile Identifier Database
- 1228 [B26] ZigBee Document 09-5499r22 – Green Power Specification

1229

1230

1231

1232

1233

1234

1235

1236

1237

1238

1239

1240

1241

1242

1243

1244

1245

1246

1247

1248

This page intentionally left blank.

1249

CHAPTER 2 APPLICATION LAYER SPECIFICATION

1250

1251 2.1 General Description

1252 The ZigBee stack architecture includes a number of layered components including the IEEE 802.15.4 Medium Access Control (MAC) layer, Physical (PHY) layer, and the ZigBee Network (NWK) layer. Each component provides an application with its own set of services and capabilities. Although this chapter may refer to other components within the ZigBee stack architecture, its primary purpose is to describe the component labeled Application (APL) shown in Figure 1.1 of “ZigBee Protocol Overview.”

1253 As shown in Figure 1.1, the ZigBee application layer consists of the APS sub-layer, the ZDO (containing the ZDO management plane), and the manufacturer-defined application objects.

1259 2.1.1 Application Support Sub-Layer

1260 The application support sub-layer (APS) provides an interface between the network layer (NWK) and the application layer (APL) through a general set of services that are used by both the ZDO and the manufacturer-defined application objects. The services are provided by two entities:

- 1263 • The APS data entity (APSDE) through the APSDE service access point (APSDE-SAP).
- 1264 • The APS management entity (APSME) through the APSME service access point (APSME-SAP).

1265 The APSDE provides the data transmission service between two or more application entities located on the same network.

1267 The APSME provides a variety of services to application objects including security services and binding of devices. It also maintains a database of managed objects, known as the APS information base (AIB).

1269 2.1.2 Application Framework

1270 The application framework in ZigBee is the environment in which application objects are hosted on ZigBee devices.

1272 Up to 254 distinct application objects can be defined, each identified by an endpoint address from 1 to 254. Two additional endpoints are defined for APSDE-SAP usage: endpoint 0 is reserved for the data interface to the ZDO, and endpoint 255 is reserved for the data interface function to broadcast data to all application objects. Endpoints 241-254 are assigned by the ZigBee Alliance and shall not be used without approval. The Green Power cluster, if implemented, shall use endpoint 242.

1277 2.1.2.1 Application Profiles

1278 Application profiles are agreements for messages, message formats, and processing actions that enable developers to create an interoperable, distributed application employing application entities that reside on separate devices. These application profiles enable applications to send commands, request data, and process commands and requests.

1282 **2.1.2.2 Clusters**

1283 Clusters are identified by a cluster identifier, which is associated with data flowing out of, or into, the device.
1284 Cluster identifiers are unique within the scope of a particular application profile.

1285 **2.1.3 ZigBee Device Objects**

1286 The ZigBee device objects (ZDO), represent a base class of functionality that provides an interface between
1287 the application objects, the device profile, and the APS. The ZDO is located between the application
1288 framework and the application support sub-layer. It satisfies common requirements of all applications op-
1289 erating in a ZigBee protocol stack. The ZDO is responsible for the following:

- 1290 • Initializing the application support sub-layer (APS), the network layer (NWK), and the Security Ser-
1291 vice Provider.
- 1292 • Assembling configuration information from the end applications to determine and implement discov-
1293 ery, security management, network management, and binding management.

1294 The ZDO presents public interfaces to the application objects in the application framework layer for control
1295 of device and network functions by the application objects. The ZDO interfaces with the lower portions of the
1296 ZigBee protocol stack, on endpoint 0, through the APSDE-SAP for data, and through the APSME-SAP and
1297 NLME-SAP for control messages. The public interface provides address management of the device, dis-
1298 covery, binding, and security functions within the application framework layer of the ZigBee protocol stack.
1299 The ZDO is fully described in clause 2.5.

1300 **2.1.3.1 Device Discovery**

1301 Device discovery is the process whereby a ZigBee device can discover other ZigBee devices. There are two
1302 forms of device discovery requests: IEEE address requests and NWK address requests. The IEEE address
1303 request is unicast to a particular device and assumes the NWK address is known. The NWK address request is
1304 broadcast and carries the known IEEE address as data payload.

1305 **2.1.3.2 Service Discovery**

1306 Service discovery is the process whereby the capabilities of a given device are discovered by other devices.
1307 Service discovery can be accomplished by issuing a query for each endpoint on a given device or by using a
1308 match service feature (either broadcast or unicast). The service discovery facility defines and utilizes various
1309 descriptors to outline the capabilities of a device.

1310 Service discovery information may also be cached in the network in the case where the device proffering a
1311 particular service may be inaccessible at the time the discovery operation takes place.

1312 **2.2 ZigBee Application Support (APS) Sub-Layer**

1313 **2.2.1 Scope**

1314 This clause specifies the portion of the application layer providing the service specification and interface to
1315 both the manufacturer-defined application objects and the ZigBee device objects. The specification defines a
1316 data service to allow the application objects to transport data, and a management service providing mecha-
1317 nisms for binding. In addition, it also defines the application support sub-layer frame format and frame-type
1318 specifications.

1319 **2.2.2 Purpose**

1320 The purpose of this clause is to define the functionality of the ZigBee application support (APS) sub-layer.
1321 This functionality is based on both the driver functionality necessary to enable correct operation of the
1322 ZigBee network layer and the functionality required by the manufacturer-defined application objects.

1323 **2.2.3 Application Support (APS) Sub-Layer Overview**

1324 The application support sub-layer provides the interface between the network layer and the application layer
1325 through a general set of services for use by both the ZigBee device object (ZDO) and the manufacturer-
1326 defined application objects. These services are offered via two entities: the data service and the man-
1327 agement service. The APS data entity (APSDE) provides the data transmission service via its associated
1328 SAP, the APSDE-SAP. The APS management entity (APSME) provides the management service via its
1329 associated SAP, the APSME-SAP, and maintains a database of managed objects known as the APS infor-
1330 mation base (AIB).

1331 **2.2.3.1 Application Support Sub-Layer Data Entity (APSDE)**

1332 The APSDE shall provide a data service to the network layer and both ZDO and application objects to enable
1333 the transport of application PDUs between two or more devices. The devices themselves must be located on
1334 the same network.

1335 The APSDE will provide the following services:

- 1336 • **Generation of the application level PDU (APDU):** The APSDE shall take an application PDU and
1337 generate an APS PDU by adding the appropriate protocol overhead.
- 1338 • **Binding:** Once two devices are bound, the APSDE shall be able to transfer a message from one bound
1339 device to the second device.
- 1340 • **Group address filtering:** The ability to filter group-addressed messages based on endpoint group
1341 membership.
- 1342 • **Reliable transport:** Increases the reliability of transactions above that available from the NWK layer
1343 alone by employing end-to-end retries.
- 1344 • **Duplicate rejection:** Messages offered for transmission will not be received more than once.
- 1345 • **Fragmentation:** Enables segmentation and reassembly of messages longer than the payload of a single
1346 NWK layer frame.

1347 **2.2.3.2 Application Support Sub-Layer Management Entity (APSME)**

1348 The APSME shall provide a management service to allow an application to interact with the stack.

1349 The APSME shall provide the ability to match two devices together based on their services and their needs.
1350 This service is called the binding service, and the APSME shall be able to construct and maintain a table to
1351 store this information.

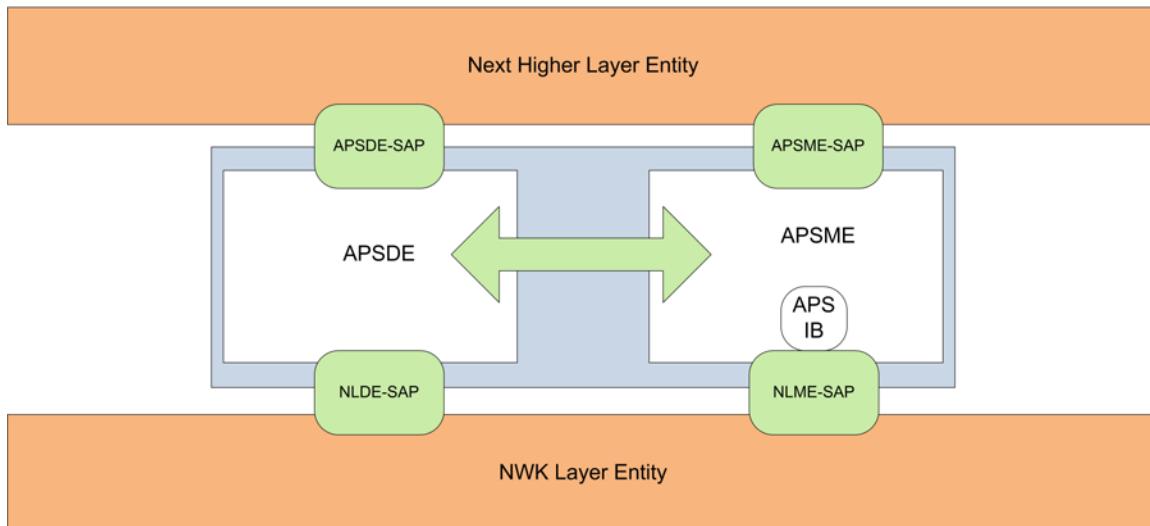
1352 In addition, the APSME will provide the following services:

- 1353 • **Binding management:** The ability to match two devices together based on their services and their
1354 needs.
- 1355 • **AIB management:** The ability to get and set attributes in the device's AIB.
- 1356 • **Security:** The ability to set up authentic relationships with other devices through the use of secure
1357 keys.
- 1358 • **Group management:** The ability to declare a single address shared by multiple devices, to add devic-
1359 es to the group, and to remove devices from the group.

1360 2.2.4 Service Specification

1361 The APS sub-layer provides an interface between a next higher layer entity (NHLE) and the NWK layer. The
1362 APS sub-layer conceptually includes a management entity called the APS sub-layer management entity
1363 (APSME). This entity provides the service interfaces through which sub-layer management functions may be
1364 invoked. The APSME is also responsible for maintaining a database of managed objects pertaining to the
1365 APS sub-layer. This database is referred to as the APS sub-layer information base (AIB). Figure 2.1 depicts
1366 the components and interfaces of the APS sub-layer.

1367 **Figure 2.1 The APS Sub-Layer Reference Model**



1368 The APS sub-layer provides two services, accessed through two service access points (SAPs). These are the
1369 APS data service, accessed through the APS sub-layer data entity SAP (APSDE-SAP), and the APS
1370 management service, accessed through the APS sub-layer management entity SAP (APSME-SAP). These two
1371 services provide the interface between the NHLE and the NWK layer, via the NLDE-SAP and, to a limited
1372 extent, NLME-SAP interfaces (see section 3.1). The NLME-SAP interface between the NWK layer and the
1373 APS sub-layer supports only the NLME-GET and NLME-SET primitives; all other NLME-SAP primitives
1374 are available only via the ZDO (see section 2.5). In addition to these external interfaces, there is also an
1375 implicit interface between the APSME and the APSDE that allows the APSME to use the APS data service.
1376

1377 2.2.4.1 APS Data Service

1378 The APS sub-layer data entity SAP (APSDE-SAP) supports the transport of application protocol data units
1379 between peer application entities. Table 2.1 lists the primitives supported by the APSDE-SAP. Each of these
1380 primitives will be discussed in the following sections.

1381 **Table 2.1 APSDE-SAP Primitives**

APSDE-SAP Primitive	Request	Confirm	Indication
APSDE-DATA	2.2.4.1.1	2.2.4.1.2	2.2.4.1.3

1382 2.2.4.1.1 APSDE-DATA.request

1383 This primitive requests the transfer of a NHLE PDU (ASDU) from the local NHLE to one or more peer
1384 NHLE entities.

1385 **2.2.4.1.1.1 Semantics of the Service Primitive**

1386 The semantics of this primitive are as follows:

1387 APSDE-DATA.request {
1388 DstAddrMode,
1389 DstAddress,
1390 DstEndpoint,
1391 ProfileId,
1392 ClusterId,
1393 SrcEndpoint,
1394 ASDULength,
1395 ASDU,
1396 TxOptions,
1397 UseAlias,
1398 AliasSrcAddr,
1399 AliasSeqNumber,
1400 RadiusCounter
1401 }

1402

1403 Table 2.2 specifies the parameters for the APSDE-DATA.request primitive. Support of the parameters
1404 UseAlias, AliasSrcAddr, and AliasSeqNumb in the APSDE-DATA.request primitive is required if Green
1405 Power feature is supported by the implementation.

1406 **Table 2.2 APSDE-DATA.request Parameters**

Name	Type	Valid Range	Description
DstAddrMode	Integer	0x00 – 0xff	The addressing mode for the destination address used in this primitive and of the APDU to be transferred. This parameter can take one of the non-reserved values from the following list: 0x00 = DstAddress and DstEndpoint not present 0x01 = 16-bit group address for DstAddress; DstEndpoint not present 0x02 = 16-bit address for DstAddress and DstEndpoint present 0x03 = 64-bit extended address for DstAddress and DstEndpoint present 0x04 – 0xff = reserved
DstAddress	Address	As specified by the DstAddrMode parameter	The individual device address or group address of the entity to which the ASDU is being transferred.

Name	Type	Valid Range	Description
DstEndpoint	Integer	0x00 – 0xff	This parameter shall be present if, and only if, the DstAddrMode parameter has a value of 0x02 or 0x03 and, if present, shall be either the number of the individual endpoint of the entity to which the ASDU is being transferred or the broadcast endpoint (0xff).
ProfileId	Integer	0x0000 – 0xffff	The identifier of the profile for which this frame is intended.
ClusterId	Integer	0x0000 – 0xffff	The identifier of the object for which this frame is intended.
SrcEndpoint	Integer	0x00 – 0xfe	The individual endpoint of the entity from which the ASDU is being transferred.
ASDULength	Integer	0x00 – 256 * (NsduLength - apscMinHeaderOverhead)	The number of octets comprising the ASDU to be transferred. The maximum length of an individual APS frame payload is given as NsduLength - <i>apscMinHeaderOverhead</i> . Assuming fragmentation is used, there can be 256 such blocks comprising a single maximum sized ASDU.
ASDU	Set of octets	-	The set of octets comprising the ASDU to be transferred.
TxOptions	Bitmap	0000 0000 – 0001 1111	The transmission options for the ASDU to be transferred. These are a bitwise OR of one or more of the following: 0x01 = Security enabled transmission 0x02 = Use NWK key 0x04 = Acknowledged transmission 0x08 = Fragmentation permitted 0x10 = Include extended nonce in APS security frame.
UseAlias	Boolean	TRUE or FALSE	The next higher layer may use the UseAlias parameter to request alias usage by NWK layer for the current frame. If the <i>UseAlias</i> parameter has a value of FALSE, meaning no alias usage, then the parameters <i>AliasSrcAddr</i> and <i>AliasSeqNumb</i> will be ignored. Otherwise, a value of TRUE denotes that the values supplied in <i>AliasSrcAddr</i> and <i>AliasSeqNumb</i> are to be used.
AliasSrcAddr	16-bit address	Any valid device address except a broadcast address	The source address to be used for this NSDU. If the <i>UseAlias</i> parameter has a value of FALSE, the <i>AliasSrcAddr</i> parameter is ignored.

Name	Type	Valid Range	Description
AliasSeqNumb	integer	0x00-0xff	The sequence number to be used for this NSDU. If the <i>UseAlias</i> parameter has a value of FALSE, the <i>AliasSeqNumb</i> parameter is ignored.
Radius	Unsigned integer	0x00-0xff	The distance, in hops, that a transmitted frame will be allowed to travel through the network.

1407 **2.2.4.1.1.2 When Generated**

1408 This primitive is generated by a local NHLE whenever a data PDU (ASDU) is to be transferred to one or
 1409 more peer NHLEs.

1410 **2.2.4.1.1.3 Effect on Receipt**

1411 On receipt of this primitive, the APS sub-layer entity begins the transmission of the supplied ASDU.

1412 If the DstAddrMode parameter is set to 0x00 and this primitive was received by the APSDE of a device
 1413 supporting a binding table, a search is made in the binding table with the endpoint and cluster identifiers
 1414 specified in the SrcEndpoint and ClusterId parameters, respectively, for associated binding table entries. If no
 1415 binding table entries are found, the APSDE issues the APSDE-DATA.confirm primitive with a status of
 1416 NO_BOUND_DEVICE. If one or more binding table entries are found, then the APSDE examines the des-
 1417 tination address information in each binding table entry. If this indicates a device itself, then the APSDE shall
 1418 issue an APSDE-DATA.indication primitive to the next higher layer with the DstEndpoint parameter set to
 1419 the destination endpoint identifier in the binding table entry. If UseAlias parameter has the value of TRUE,
 1420 the supplied value of the AliasSrcAddr shall be used for the SrcAddress parameter of the
 1421 APSDE-DATA.indication primitive. Otherwise if the binding table entries do not indicate the device itself,,
 1422 the APSDE constructs the APDU with the endpoint information from the binding table entry, if present, and
 1423 uses the destination address information from the binding table entry when transmitting the frame via the
 1424 NWK layer. If more than one binding table entry is present, then the APSDE processes each binding table
 1425 entry as described above; until no more binding table entries remain. If this primitive was received by the
 1426 APSDE of a device that does not support a binding table, the APSDE issues the APSDE-DATA.confirm
 1427 primitive with a status of
 1428 NOT_SUPPORTED.

1429 If the DstAddrMode parameter is set to 0x03, the DstAddress parameter contains an extended 64-bit IEEE
 1430 address and must first be mapped to a corresponding 16-bit NWK address by using the nwkAddressMap
 1431 attribute of the NIB (see Table 3.43). If a corresponding 16-bit NWK address could not be found, the APSDE
 1432 issues the APSDE-DATA.confirm primitive with a status of NO_SHORT_ADDRESS. If a corresponding
 1433 16-bit NWK address is found, it will be used in the invocation of the NLDE-DATA.request primitive and the
 1434 value of the DstEndpoint parameter will be placed in the resulting APDU. The delivery mode sub-field of the
 1435 frame control field of the APS header shall have a value of 0x00 in this case.

1436 If the DstAddrMode parameter has a value of 0x01, indicating group addressing, the DstAddress parameter
 1437 will be interpreted as a 16-bit group address. This address will be placed in the group address field of the APS
 1438 header, the DstEndpoint parameter will be ignored, and the destination endpoint field will be omitted from
 1439 the APS header. The delivery mode sub-field of the frame control field of the APS header shall have a value
 1440 of 0x03 in this case.

1441 If the DstAddrMode parameter is set to 0x02, the DstAddress parameter contains a 16-bit NWK address, and
 1442 the DstEndpoint parameter is supplied. The next higher layer should only employ DstAddrMode of 0x02 in
 1443 cases where the destination NWK address is employed for immediate application responses and the NWK
 1444 address is not retained for later data transmission requests.

1445 The application may limit the number of hops a transmitted frame is allowed to travel through the network by
 1446 setting the RadiusCounter parameter of the NLDE-DATA.request primitive to a non-zero value.

1447 If the DstAddrMode parameter has a value of 0x01, indicating group addressing, or the DstAddrMode pa-
1448 rameter has a value of 0x00 and the corresponding binding table entry contains a group address, then the
1449 APSME will check the value of the nwkUseMulticast attribute of the NIB (see Table 3.44). If this attribute
1450 has a value of FALSE, then the delivery mode sub-field of the frame control field of the resulting APDU will
1451 be set to 0b11, the 16-bit address of the destination group will be placed in the group address field of the APS
1452 header of the outgoing frame, and the NSDU frame will be transmitted as a broadcast. A value of 0xffffd, that
1453 is, the broadcast to all devices for which macRxOnWhenIdle = TRUE, will be supplied for the DstAddr
1454 parameter of the NLDE-DATA.request that is used to transmit the frame. If the *nwkUseMulticast* attribute
1455 has a value of TRUE, then the outgoing frame will be transmitted using NWK layer multicast, with the de-
1456 livery mode sub-field of the frame control field of the APDU set to 0b10, the destination endpoint field set to
1457 0xff, and the group address not placed in the APS header.

1458 The parameters UseAlias, AliasSrcAddr and AliasSeqNumb shall be used in the invocation of the
1459 NLDE-DATA.request primitive.

1460 If the UseAlias parameter has the value of TRUE, and the Acknowledged transmission field of the TxOptions
1461 parameter is set to 0b1, then the APSDE issues the APSDE-DATA.confirm primitive with a status of
1462 NOT_SUPPORTED.

1463 If the TxOptions parameter specifies that secured transmission is required, the APS sub-layer shall use the
1464 security service provider (see section 4.2.3) to secure the ASDU. The security processing shall always be
1465 performed using device's own extended 64-bit IEEE address and the OutgoingFrameCounter attribute as
1466 stored in apsDeviceKeyPairSet attribute of the AIB for the entity indicated by the DstAddress parameter, and
1467 those values shall be put into the auxiliary APS header of the frame, even if UseAlias parameter has a value of
1468 TRUE. If the security processing fails, the APSDE shall issue the APSDE-DATA.confirm primitive with a
1469 status of SECURITY_FAIL.

1470 The APSDE transmits the constructed frame by issuing the NLDE-DATA.request primitive to the NWK
1471 layer. When the APSDE has completed all operations related to this transmission request, including trans-
1472 mitting frames as required, any retransmissions, and the receipt or timeout of any acknowledgements, then
1473 the APSDE shall issue the APSDE-DATA.confirm primitive (see section 2.2.4.1.2). If one or more
1474 NLDE-DATA.confirm primitives failed, then the Status parameter shall be set to that received from the
1475 NWK layer. Otherwise, if one or more APS acknowledgements were not correctly received, then the Status
1476 parameter shall be set to NO_ACK. If the ASDU was successfully transferred to all intended targets, then the
1477 Status parameter shall be set to SUCCESS.

1478 If NWK layer multicast is being used, the NonmemberRadius parameter of the NLDE-DATA.request
1479 primitive shall be set to *apsNonmemberRadius*.

1480 The APSDE will ensure that route discovery is always enabled at the network layer by setting the Dis-
1481 coverRoute parameter of the NLDE-DATA.request primitive to 0x01, each time it is issued.

1482 If the ASDU to be transmitted is larger than will fit in a single frame and fragmentation is not possible, then
1483 the ASDU is not transmitted and the APSDE shall issue the APSDE-DATA.confirm primitive with a status
1484 of ASDU_TOO_LONG. Fragmentation is not possible if either an acknowledged transmission is not re-
1485 quested, or if the fragmentation permitted flag in the TxOptions field is set to 0, or if the ASDU is too large to
1486 be handled by the APSDE.

1487 If the ASDU to be transmitted is larger than will fit in a single frame, an acknowledged transmission is re-
1488 quested, and the fragmentation permitted flag of the TxOptions field is set to 1, and the ASDU is not too large
1489 to be handled by the APSDE, then the ASDU shall be fragmented across multiple APDUs, as described in
1490 section 2.2.8.4.5. Transmission and security processing where requested, shall be carried out for each indi-
1491 vidual APDU independently. Note that fragmentation shall not be used unless relevant higher-layer docu-
1492 mentation and/or interactions explicitly indicate that fragmentation is permitted for the frame being sent, and
1493 that the other end is able to receive the fragmented transmission, both in terms of number of blocks and total
1494 transmission size.

1495 **2.2.4.1.2 APSDE-DATA.confirm**

1496 The primitive reports the results of a request to transfer a data PDU (ASDU) from a local NHLE to one or
1497 more peer NHLEs.

1498 **2.2.4.1.2.1 Semantics of the Service Primitive**

1499 This semantics of this primitive are as follows:

1500 APSDE-DATA.confirm {
1501 DstAddrMode,
1502 DstAddress,
1503 DstEndpoint,
1504 SrcEndpoint,
1505 Status,
1506 TxTime
1507 }

1508

1509 Table 2.3 specifies the parameters for the APSDE-DATA.confirm primitive.

1510 **Table 2.3 APSDE-DATA.confirm Parameters**

Name	Type	Valid Range	Description
DstAddrMode	Integer	0x00 – 0xff	The addressing mode for the destination address used in this primitive and of the APDU to be transferred. This parameter can take one of the non-reserved values from the following list: 0x00 = DstAddress and DstEndpoint not present 0x01 = 16-bit group address for DstAddress; DstEndpoint not present 0x02 = 16-bit address for DstAddress and DstEndpoint present 0x03= 64-bit extended address for DstAddress and DstEndpoint present 0x04 – 0xff = reserved
DstAddress	Address	As specified by the DstAddrMode parameter	The individual device address or group address of the entity to which the ASDU is being transferred.
DstEndpoint	Integer	0x00 – 0xff	This parameter shall be present if, and only if, the DstAddrMode parameter has a value of 0x02 or 0x03 and, if present, shall be the number of the individual endpoint of the entity to which the ASDU is being transferred.
SrcEndpoint	Integer	0x00 – 0xfe	The individual endpoint of the entity from which the ASDU is being transferred.

Name	Type	Valid Range	Description
Status	Enumeration	SUCCESS, NO_SHORT_ADDRESS, NO_BOUND_DEVICE, SECURITY_FAIL, NO_ACK, ASDU_TOO_LONG or any status values returned from the NLDE-DATA.confirm primitive	The status of the corresponding request.
TxTime	Integer	Implementation specific	A time indication for the transmitted packet based on the local clock, as provided by the NWK layer.

1511 **2.2.4.1.2.2 When Generated**

1512 This primitive is generated by the local APS sub-layer entity in response to an APSDE-DATA.request
1513 primitive. This primitive returns a status of either SUCCESS, indicating that the request to transmit was
1514 successful, or an error code of NO_SHORT_ADDRESS, NO_BOUND_DEVICE, SECURITY_FAIL,
1515 ASDU_TOO_LONG, or any status values returned from the NLDE-DATA.confirm primitive. The reasons
1516 for these status values are fully described in section 2.2.4.1.3.

1517 **2.2.4.1.2.3 Effect on Receipt**

1518 On receipt of this primitive, the next higher layer of the initiating device is notified of the result of its request
1519 to transmit. If the transmission attempt was successful, the Status parameter will be set to SUCCESS. Oth-
1520 erwise, the Status parameter will indicate the error.

1521 **2.2.4.1.3 APSDE-DATA.indication**

1522 This primitive indicates the transfer of a data PDU (ASDU) from the APS sub-layer to the local application
1523 entity.

1524 **2.2.4.1.3.1 Semantics of the Service Primitive**

1525 The semantics of this primitive are as follows:

1526 APSDE-DATA.indication {
1527 DstAddrMode,
1528 DstAddress,
1529 DstEndpoint,
1530 SrcAddrMode,
1531 SrcAddress,
1532 SrcEndpoint,
1533 ProfileId,
1534 ClusterId,
1535 asduLength,
1536 asdu,
1537 Status,
1538 SecurityStatus,
1539 LinkQuality,
1540 RxTime
1541 }

1542

1543 Table 2.4 specifies the parameters for the APSDE-DATA.indication primitive.

1544

Table 2.4 APSDE-DATA.indication Parameters

Name	Type	Valid Range	Description
DstAddrMode	Integer	0x00 - 0xff	The addressing mode for the destination address used in this primitive and of the APDU that has been received. This parameter can take one of the non-reserved values from the following list: 0x00 = reserved 0x01 = 16-bit group address for DstAddress; DstEndpoint not present 0x02 = 16-bit address for DstAddress and DstEndpoint present 0x03 = 64-bit extended address for DstAddress and DstEndpoint present. 0x04 = 64-bit extended address for DstAddress, but DstEndpoint NOT present. 0x05 – 0xff = reserved
DstAddress	Address	As specified by the DstAddrMode parameter	The individual device address or group address to which the ASDU is directed.

Name	Type	Valid Range	Description
DstEndpoint	Integer	0x00 – 0xfe	The target endpoint on the local entity to which the ASDU is directed.
SrcAddrMode	Integer	0x00 – 0xff	The addressing mode for the source address used in this primitive and of the APDU that has been received. This parameter can take one of the non-reserved values from the following list: 0x00 = reserved 0x01 = reserved 0x02 = 16-bit short address for SrcAddress and SrcEndpoint present 0x03 = 64-bit extended address for SrcAddress and SrcEndpoint present 0x04 = 64-bit extended address for SrcAddress, but SrcEndpoint NOT present. 0x05 – 0xff = reserved
SrcAddress	Address	As specified by the SrcAddrMode parameter	The individual device address of the entity from which the ASDU has been received.
SrcEndpoint	Integer	0x00 – 0xfe	The number of the individual endpoint of the entity from which the ASDU has been received.
ProfileId	Integer	0x0000 - 0xffff	The identifier of the profile from which this frame originated.
ClusterId	Integer	0x0000-0xffff	The identifier of the received object.
asduLength	Integer		The number of octets comprising the ASDU being indicated by the APSDE.
asdu	Set of octets	-	The set of octets comprising the ASDU being indicated by the APSDE.
Status	Enumeration	SUCCESS, DEFRAG_UNSUPPORTED, DEFRAG_DEFERRED or any status returned from the security processing of the frame	The status of the incoming frame processing.
SecurityStatus	Enumeration	UNSECURED, SECURED_NWK_KEY,	UNSECURED if the ASDU was received without any security.

Name	Type	Valid Range	Description
		SECURED_LINK_KEY	SECURED_NWK_KEY if the received ASDU was secured with the NWK key. SECURED_LINK_KEY if the ASDU was secured with a link key.
LinkQuality	Integer	0x00 - 0xff	The link quality indication delivered by the NLDE.
RxTime	Integer	Implementation specific	A time indication for the received packet based on the local clock, as provided by the NWK layer.

1545 **2.2.4.1.3.2 When Generated**

1546 This primitive is generated by the APS sub-layer and issued to the next higher layer on receipt of an appro-
 1547 priately addressed data frame from the local NWK layer entity or following receipt of an APSDE-DATA.
 1548 request in which the DstAddrMode parameter was set to 0x00 and the binding table entry has directed the
 1549 frame to the device itself. If the frame control field of the ASDU header indicates that the frame is secured,
 1550 security processing shall be done as specified in section 4.2.3.

1551 This primitive is generated by the APS sub-layer entity and issued to the next higher layer entity on receipt of
 1552 an appropriately addressed data frame from the local network layer entity, via the NLDE-DATA.indication
 1553 primitive.

1554 If the frame control field of the APDU header indicates that the frame is secured, then security processing
 1555 must be undertaken as specified in section 4.2.3. If the security processing fails, the APSDE sets the Status
 1556 parameter to the security error code returned from the security processing.

1557 If the frame is not secured or the security processing was successful, the APSDE must check for the frame
 1558 being fragmented. If the extended header is included in the APDU header and the fragmentation sub-field of
 1559 the extended frame control field indicates that the frame is fragmented but this device does not support
 1560 fragmentation, the APSDE sets the Status parameter to DEFrag_UNSUPPORTED. If the extended header
 1561 is included in the APDU header, the fragmentation sub-field of the extended frame control field indicates that
 1562 the frame is fragmented and the device supports fragmentation, but is not currently able to defragment the
 1563 frame, the APSDE sets the Status parameter to DEFrag_DEFERRED.

1564 Under all other circumstances, the APSDE sets the Status parameter to SUCCESS.

1565 If the Status parameter is not set to SUCCESS, the APSDE sets the ASDULength parameter to 0 and the
 1566 ASDU parameter to the null set of bytes.

1567 The APS sub-layer entity shall attempt to map the source address from the received frame to its corre-
 1568 sponding extended 64-bit IEEE address by using the nwkAddressMap attribute of the NIB (see Table 3.43).
 1569 If a corresponding 64-bit IEEE address was found, the APSDE issues this primitive with the SrcAddrMode
 1570 parameter set to 0x03 and the SrcAddress parameter set to the corresponding 64-bit IEEE address. If a cor-
 1571 responding 64-bit IEEE address was not found, the APSDE issues this primitive with the SrcAddrMode
 1572 parameter set to 0x02, and the SrcAddress parameter set to the 16-bit source address as contained in the re-
 1573 ceived frame.

1574 **2.2.4.1.3.3 Effect on Receipt**

1575 On receipt of this primitive, the next higher layer is notified of the arrival of data at the device.

1576 **2.2.4.2 APS Management Service**

1577 The APS management entity SAP (APSME-SAP) supports the transport of management commands between
1578 the next higher layer and the APSME. Table 2.5 summarizes the primitives supported by the APSME through
1579 the APSME-SAP interface. See the following sections for more details on the individual primitives.

1580 **Table 2.5 Summary of the Primitives Accessed Through the APSME-SAP**

Name	Request	Indication	Response	Confirm
APSME-BIND	2.2.4.3.1			2.2.4.3.2
APSME-UNBIND	2.2.4.3.3			2.2.4.3.4
APSME-GET	2.2.4.4.1			2.2.4.4.2
APSME-SET	2.2.4.4.3			2.2.4.4.4
APSME-ADD-GROUP	2.2.4.5.1			2.2.4.5.2
APSME-REMOVE-GROUP	2.2.4.5.3			2.2.4.5.4
APSME-REMOVE-ALL-GROUPS	2.2.4.5.5			2.2.4.5.6

1581 **2.2.4.3 Binding Primitives**

1582 This set of primitives defines how the next higher layer of a device can add (commit) a binding record to its
1583 local binding table or remove a binding record from its local binding table.

1584 Only a device supporting a binding table or a binding table cache may process these primitives. If any other
1585 device receives these primitives from their next higher layer, the primitives should be rejected.

1586 **2.2.4.3.1 APSME-BIND.request**

1587 This primitive allows the next higher layer to request to bind two devices together, or to bind a device to a
1588 group, by creating an entry in its local binding table, if supported.

1589 **2.2.4.3.1.1 Semantics of the Service Primitive**

1590 The semantics of this primitive are as follows:

```
1591     APSME-BIND.request      {  
1592         SrcAddr,  
1593         SrcEndpoint,  
1594         ClusterId,  
1595         DstAddrMode,  
1596         DstAddr,  
1597         DstEndpoint  
1598     }
```

1599

1600 Table 2.6 specifies the parameters for the APSME-BIND.request primitive.

1601

Table 2.6 APSME-BIND.request Parameters

Name	Type	Valid Range	Description
SrcAddr	IEEE address	A valid 64-bit IEEE address	The source IEEE address for the binding entry.
SrcEndpoint	Integer	0x01 – 0xfe	The source endpoint for the binding entry.
ClusterId	Integer	0x0000 – 0xffff	The identifier of the cluster on the source device that is to be bound to the destination device.
DstAddrMode	Integer	0x00 – 0xff	The addressing mode for the destination address used in this primitive. This parameter can take one of the non-reserved values from the following list: 0x00 = reserved 0x01 = 16-bit group address for DstAddr and DstEndpoint not present 0x02 = reserved 0x03 = 64-bit extended address for DstAddr and DstEndpoint present 0x04 – 0xff = reserved
DstAddr	Address	As specified by the DstAddrMode parameter	The destination address for the binding entry.
DstEndpoint	Integer	0x01 – 0xff	This parameter will be present only if the DstAddrMode parameter has a value of 0x03 and, if present, will be the destination endpoint for the binding entry.

1602

2.2.4.3.1.2 When Generated

1603

This primitive is generated by the next higher layer and issued to the APS sub-layer in order to instigate a binding operation on a device that supports a binding table.

1604

2.2.4.3.1.3 Effect on Receipt

1605

On receipt of this primitive by a device that is not currently joined to a network, or by a device that does not support a binding table, or if any of the parameters has a value which is out of range, the APSME issues the APSME-BIND.confirm primitive with the Status parameter set to ILLEGAL_REQUEST.

1606

If the APS sub-layer on a device that supports a binding table receives this primitive from the NHLE, the APSME attempts to create the specified entry directly in its binding table. If the entry could be created, the APSME issues the APSME-BIND.confirm primitive with the Status parameter set to SUCCESS. If the entry could not be created due to a lack of capacity in the binding table, the APSME issues the APSME-BIND.confirm primitive with the Status parameter set to TABLE_FULL.

1607

2.2.4.3.2 APSME-BIND.confirm

1608

This primitive allows the next higher layer to be notified of the results of its request to bind two devices together, or to bind a device to a group.

1617 **2.2.4.3.2.1 Semantics of the Service Primitive**

1618 The semantics of this primitive are as follows:

1619 APSME-BIND.confirm {
1620 Status,
1621 SrcAddr,
1622 SrcEndpoint,
1623 ClusterId,
1624 DstAddrMode,
1625 DstAddr,
1626 DstEndpoint
1627 }

1628

1629 Table 2.7 specifies the parameters for the APSME-BIND.confirm primitive.

1630 **Table 2.7 APSME-BIND.confirm Parameters**

Name	Type	Valid Range	Description
Status	Enumeration	SUCCESS, ILLEGAL_REQUEST, TA- BLE_FULL, NOT_SUPPORTED	The results of the binding request.
SrcAddr	IEEE address	A valid 64-bit IEEE address	The source IEEE address for the binding entry.
SrcEndpoint	Integer	0x01 – 0xfe	The source endpoint for the binding entry.
ClusterId	Integer	0x0000 – 0xffff	The identifier of the cluster on the source de- vice that is to be bound to the destination de- vice.
DstAddrMode	Integer	0x00 – 0xff	The addressing mode for the destination ad- dress used in this primitive. This parameter can take one of the non-reserved values from the following list: 0x00 = reserved 0x01 = 16-bit group address for DstAddr and DstEndpoint not present 0x02 = reserved 0x03 = 64-bit extended address for DstAddr and DstEndpoint present 0x04 – 0xff = reserved
DstAddr	Address	As specified by the DstAddrMode parameter	The destination address for the binding entry.

Name	Type	Valid Range	Description
DstEndpoint	Integer	0x01 – 0xff	This parameter will be present only if the DstAddrMode parameter has a value of 0x03 and, if present, will be the destination endpoint for the binding entry.

1631 **2.2.4.3.2.2 When Generated**

1632 This primitive is generated by the APSME and issued to its NHLE in response to an APSME-BIND.request
 1633 primitive. If the request was successful, the Status parameter will indicate a successful bind request. Otherwise,
 1634 the Status parameter indicates an error code of NOT_SUPPORTED, ILLEGAL_REQUEST or
 1635 TABLE_FULL.

1636 **2.2.4.3.2.3 Effect on Receipt**

1637 On receipt of this primitive, the next higher layer is notified of the results of its bind request. If the bind
 1638 request was successful, the Status parameter is set to SUCCESS. Otherwise, the Status parameter indicates
 1639 the error.

1640 **2.2.4.3.3 APSME-UNBIND.request**

1641 This primitive allows the next higher layer to request to unbind two devices, or to unbind a device from a
 1642 group, by removing an entry in its local binding table, if supported.

1643 **2.2.4.3.3.1 Semantics of the Service Primitive**

1644 The semantics of this primitive are as follows:

1645 APSME-UNBIND.request	{
1646	SrcAddr,
1647	SrcEndpoint,
1648	ClusterId,
1649	DstAddrMode,
1650	DstAddr,
1651	DstEndpoint
1652	}

1653
 1654 Table 2.8 specifies the parameters for the APSME-UNBIND.request primitive.

1655 **Table 2.8 APSME-UNBIND.request Parameters**

Name	Type	Valid Range	Description
SrcAddr	IEEE address	A valid 64-bit IEEE address	The source IEEE address for the binding entry.
SrcEndpoint	Integer	0x01 – 0xfe	The source endpoint for the binding entry.
ClusterId	Integer	0x0000 – 0xffff	The identifier of the cluster on the source device that is bound to the destination device.

Name	Type	Valid Range	Description
DstAddrMode	Integer	0x00 – 0xff	The addressing mode for the destination address used in this primitive. This parameter can take one of the non-reserved values from the following list: 0x00 = reserved 0x01 = 16-bit group address for DstAddr and DstEndpoint not present 0x02 = reserved 0x03 = 64-bit extended address for DstAddr and DstEndpoint present 0x04 – 0xff = reserved
DstAddr	Address	As specified by the DstAddrMode parameter	The destination address for the binding entry.
DstEndpoint	Integer	0x01 – 0xff	This parameter will be present only if the DstAddrMode parameter has a value of 0x03 and, if present, will be the destination endpoint for the binding entry.

1656 **2.2.4.3.3.2 When Generated**

1657 This primitive is generated by the next higher layer and issued to the APS sub-layer in order to instigate an
 1658 unbind operation on a device that supports a binding table.

1659 **2.2.4.3.3.3 Effect on Receipt**

1660 On receipt of this primitive by a device that is not currently joined to a network, or by a device that does not
 1661 support a binding table, or if any of the parameters has a value which is out of range, the APSME issues the
 1662 APSME-UNBIND.confirm primitive with the Status parameter set to ILLEGAL_REQUEST.

1663 If the APS on a device that supports a binding table receives this primitive from the NHLE, the APSME
 1664 searches for the specified entry in its binding table. If the entry exists, the APSME removes the entry and
 1665 issues the APSME-UNBIND.confirm (see section 2.2.4.3.4) primitive with the Status parameter set to
 1666 SUCCESS. If the entry could not be found, the APSME issues the APSME-UNBIND.confirm primitive with
 1667 the Status parameter set to INVALID_BINDING.

1668 **2.2.4.3.4 APSME-UNBIND.confirm**

1669 This primitive allows the next higher layer to be notified of the results of its request to unbind two devices, or
 1670 to unbind a device from a group.

1671 **2.2.4.3.4.1 Semantics of the Service Primitive**

1672 The semantics of this primitive are as follows:

1673 APSME-UNBIND.confirm {
1674 Status,
1675 SrcAddr,
1676 SrcEndpoint,
1677 ClusterId,
1678 DstAddrMode,
1679 DstAddr,
1680 DstEndpoint
1681 }

1682 Table 2.9 specifies the parameters for the APSME-UNBIND.confirm primitive.

1683 **Table 2.9 APSME-UNBIND.confirm Parameters**

Name	Type	Valid Range	Description
Status	Enumeration	SUCCESS, ILLEGAL_REQUEST, INVALID_BINDING	The results of the unbind request.
SrcAddr	IEEE address	A valid 64-bit IEEE address	The source IEEE address for the binding entry.
SrcEndpoint	Integer	0x01 – 0xfe	The source endpoint for the binding entry.
ClusterId	Integer	0x0000 – 0xffff	The identifier of the cluster on the source device that is bound to the destination device.
DstAddrMode	Integer	0x00 – 0xff	The addressing mode for the destination address used in this primitive. This parameter can take one of the non-reserved values from the following list: 0x00 = reserved 0x01 = 16-bit group address for DstAddr and DstEndpoint not present 0x02 = reserved 0x03 = 64-bit extended address for DstAddr and DstEndpoint present 0x04 – 0xff = reserved
DstAddr	Address	As specified by the DstAddr-Mode parameter	The destination address for the binding entry.
DstEndpoint	Integer	0x01 – 0xff	The destination endpoint for the binding

Name	Type	Valid Range	Description
			entry.

1684 **2.2.4.3.4.2 When Generated**

1685 This primitive is generated by the APSME and issued to its NHLE in response to an APSME-UNBIND.
1686 request primitive. If the request was successful, the Status parameter will indicate a successful unbind re-
1687 quest. Otherwise, the Status parameter indicates an error code of ILLEGAL_REQUEST, or INVA-
1688 LID_BINDING.

1689 **2.2.4.3.4.3 Effect on Receipt**

1690 On receipt of this primitive, the next higher layer is notified of the results of its unbind request. If the unbind
1691 request was successful, the Status parameter is set to SUCCESS. Otherwise, the Status parameter indicates
1692 the error.

1693 **2.2.4.4 Information Base Maintenance**

1694 This set of primitives defines how the next higher layer of a device can read and write attributes in the AIB.

1695 **2.2.4.4.1 APSME-GET.request**

1696 This primitive allows the next higher layer to read the value of an attribute from the AIB.

1697 **2.2.4.4.1.1 Semantics of the Service Primitive**

1698 The semantics of this primitive are as follows:

```
1699      APSME-GET.request      {  
1700                              AIBAttribute  
1701                              }  


---


```

1702 Table 2.10 specifies the parameters for this primitive.

1703 **Table 2.10 APSME-GET.request Parameters**

Name	Type	Valid Range	Description
AIBAttribute	Integer	See Table 2.24	The identifier of the AIB attribute to read.

1705 **2.2.4.4.1.2 When Generated**

1706 This primitive is generated by the next higher layer and issued to its APSME in order to read an attribute from
1707 the AIB.

1708 **2.2.4.4.1.3 Effect on Receipt**

1709 On receipt of this primitive, the APSME attempts to retrieve the requested AIB attribute from its database. If
1710 the identifier of the AIB attribute is not found in the database, the APSME issues the APSME-GET.confirm
1711 primitive with a status of UNSUPPORTED_ATTRIBUTE.

1712 If the requested AIB attribute is successfully retrieved, the APSME issues the APSME-GET.confirm primitive
1713 with a status of SUCCESS such that it contains the AIB attribute identifier and value.

1714 **2.2.4.4.2 APSME-GET.confirm**

1715 This primitive reports the results of an attempt to read the value of an attribute from the AIB.

1716 **2.2.4.4.2.1 Semantics of the Service Primitive**

1717 The semantics of this primitive are as follows:

1718 APSME-GET.confirm {
1719 Status,
1720 AIBAttribute,
1721 AIBAttributeLength,
1722 AIBAttributeValue
1723 }

1724

1725 Table 2.11 specifies the parameters for this primitive.

1726 **Table 2.11 APSME-GET.confirm Parameters**

Name	Type	Valid Range	Description
Status	Enumeration	SUCCESS or UNSUPPORTED_ATTRIBUTE	The results of the request to read an AIB attribute value.
AIBAttribute	Integer	See Table 2.24	The identifier of the AIB attribute that was read.
AIBAttributeLength	Integer	0x0001 - 0xffff	The length, in octets, of the attribute value being returned.
AIBAttributeValue	Various	Attribute specific (see Table 2.24)	The value of the AIB attribute that was read.

1727 **2.2.4.4.2.2 When Generated**

1728 This primitive is generated by the APSME and issued to its next higher layer in response to an
1729 APSME-GET.request primitive. This primitive returns a status of SUCCESS, indicating that the request to
1730 read an AIB attribute was successful, or an error code of UNSUPPORTED_ATTRIBUTE. The reasons for
1731 these status values are fully described in section 2.2.4.4.1.3.

1732 **2.2.4.4.2.3 Effect on Receipt**

1733 On receipt of this primitive, the next higher layer is notified of the results of its request to read an AIB at-
1734 tribute. If the request to read an AIB attribute was successful, the Status parameter will be set to SUCCESS.
1735 Otherwise, the Status parameter indicates the error.

1736 **2.2.4.4.3 APSME-SET.request**

1737 This primitive allows the next higher layer to write the value of an attribute into the AIB.

1738 **2.2.4.4.3.1 Semantics of the Service Primitive**

1739 The semantics of this primitive are as follows:

1740 APSME-SET.request {
1741 AIBAttribute,
1742 AIBAttributeLength,
1743 AIBAttributeValue
1744 }

1745

1746 Table 2.12 specifies the parameters for this primitive.

1747 **Table 2.12 APSME-SET.request Parameters**

Name	Type	Valid Range	Description
AIBAttribute	Integer	See Table 2.24	The identifier of the AIB attribute to be written.
AIBAttributeLength	Integer	0x0000 - 0xffff	The length, in octets, of the attribute value being set.
AIBAttributeValue	Various	Attribute specific (see Table 2.24).	The value of the AIB attribute that should be written.

1748 **2.2.4.4.3.2 When Generated**

1749 This primitive is to be generated by the next higher layer and issued to its APSME in order to write the value
1750 of an attribute in the AIB.

1751 **2.2.4.4.3.3 Effect on Receipt**

1752 On receipt of this primitive, the APSME attempts to write the given value to the indicated AIB attribute in its
1753 database. If the AIBAttribute parameter specifies an attribute that is not found in the database, the APSME
1754 issues the APSME-SET.confirm primitive with a status of UNSUPPORTED_ATTRIBUTE. If the
1755 AIBAttributeValue parameter specifies a value that is outside the valid range for the given attribute, the
1756 APSME issues the APSME-SET.confirm primitive with a status of INVALID_PARAMETER.

1757 If the requested AIB attribute is successfully written, the APSME issues the APSME-SET.confirm primitive
1758 with a status of SUCCESS.

1759 **2.2.4.4.4 APSME-SET.confirm**

1760 This primitive reports the results of an attempt to write a value to an AIB attribute.

1761 **2.2.4.4.4.1 Semantics of the Service Primitive**

1762 The semantics of this primitive are as follows:

1763 APSME-SET.confirm {
1764 Status,
1765 AIBAttribute
1766 }

1767

1768 Table 2.13 specifies the parameters for this primitive.

1769

Table 2.13 APSME-SET.confirm Parameters

Name	Type	Valid Range	Description
Status	Enumeration	SUCCESS, INVALID_PARAMETER or UNSUPPORTED_ATTRIBUTE	The result of the request to write the AIB Attribute.
AIBAttribute	Integer	See Table 2.24	The identifier of the AIB attribute that was written.

1770

2.2.4.4.4.2 When Generated

1771
1772
1773
1774
1775

This primitive is generated by the APSME and issued to its next higher layer in response to an APSME-SET.request primitive. This primitive returns a status of either SUCCESS, indicating that the requested value was written to the indicated AIB attribute, or an error code of INVALID_PARAMETER or UNSUPPORTED_ATTRIBUTE. The reasons for these status values are completely described in section 2.2.4.4.3.3.

1776

2.2.4.4.4.3 Effect on Receipt

1777
1778
1779

On receipt of this primitive, the next higher layer is notified of the results of its request to write the value of a AIB attribute. If the requested value was written to the indicated AIB attribute, the Status parameter will be set to SUCCESS. Otherwise, the Status parameter indicates the error.

1780

2.2.4.5 Group Management

1781
1782

This set of primitives allows the next higher layer to manage group membership for endpoints on the current device by adding and removing entries in the group table.

1783

2.2.4.5.1 APSME-ADD-GROUP.request

1784
1785

This primitive allows the next higher layer to request that group membership for a particular group be added for a particular endpoint.

1786

2.2.4.5.1.1 Semantics of the Service Primitive

1787

The semantics of this primitive are as follows:

```
APSME-ADD-GROUP.request {  
    GroupAddress,  
    Endpoint  
}
```

1792

1793 Table 2.14 specifies the parameters for this primitive.

1794 **Table 2.14 APSME-ADD-GROUP.request Parameters**

Name	Type	Valid Range	Description
GroupAddress	16-bit group address	0x0000 - 0xffff	The 16-bit address of the group being added.
Endpoint	Integer	0x01 - 0xfe	The endpoint to which the given group is being added.

1795 **2.2.4.5.1.2 When Generated**

1796 This primitive is generated by the next higher layer when it wants to add membership in a particular group to
1797 an endpoint, so that frames addressed to the group will be delivered to that endpoint in the future.

1798 **2.2.4.5.1.3 Effect on Receipt**

1799 If, on receipt of this primitive, the GroupAddress parameter is found to be outside the valid range, then the
1800 APSME will issue the APSME-ADD-GROUP.confirm primitive to the next higher layer with a status value
1801 of INVALID_PARAMETER. Similarly, if the Endpoint parameter has a value which is out of range or else
1802 enumerates an endpoint that is not implemented on the current device, the APSME will issue the
1803 APSME-ADD-GROUP.confirm primitive with a Status of INVALID_PARAMETER.

1804 After checking the parameters as described above, the APSME will check the group table to see if an entry
1805 already exists containing the values given by the GroupAddress and Endpoint parameters. If such an entry
1806 already exists in the table then the APSME will issue the APSME-ADD-GROUP.confirm primitive to the
1807 next higher layer with a status value of SUCCESS. If there is no such entry and there is space in the table for
1808 another entry then the APSME will add a new entry to the group table with the values given by the
1809 GroupAddress and Endpoint parameters. After the entry is added to the APS group table, and if the NWK
1810 layer is maintaining a group table, then the APSME ensures that the corresponding NWK group table is
1811 consistent by issuing the NLME-SET.request primitive, for the *nwkGroupIDTable* attribute, with the list of
1812 group addresses contained in the group table of the APS sub-layer. Once both tables are consistent, the
1813 APSME issues the APSME-ADD-GROUP.confirm primitive to the next higher layer with a status value of
1814 SUCCESS. If no entry for the given GroupAddress and Endpoint is present but there is no room in the group
1815 table for another entry, then the APSME will issue the APSME-ADD-GROUP.confirm primitive to the next
1816 higher layer with a status value of TABLE_FULL.

1817 **2.2.4.5.2 APSME-ADD-GROUP.confirm**

1818 This primitive allows the next higher layer to be informed of the results of its request to add a group to an
1819 endpoint.

1820 **2.2.4.5.2.1 Semantics of the Service Primitive**

1821 The semantics of the service primitive are as follows:

1822 APSME-ADD-GROUP.confirm {
1823 Status,
1824 GroupAddress,
1825 Endpoint
1826 }

1828 Table 2.15 specifies the parameters for this primitive.

1829 **Table 2.15 APSME-ADD-GROUP.confirm Parameters**

Name	Type	Valid Range	Description
Status	Enumeration	SUCCESS, INVALID_PARAMETER or TABLE_FULL	The status of the request to add a group.
GroupAddress	16-bit group address	0x0000 - 0xffff	The 16-bit address of the group being added.
Endpoint	Integer	0x01 - 0xfe	The endpoint to which the given group is being added.

1830 **2.2.4.5.2.2 When Generated**

1831 This primitive is generated by the APSME and issued to the next higher layer in response to an
1832 APMSE-ADD-GROUP.request primitive. If the APMSE-ADD-GROUP.request was successful, then the
1833 Status parameter value will be SUCCESS. If one of the parameters of the APMSE-ADD-GROUP.request
1834 primitive had an invalid value, then the status value will be set to INVALID_PARAMETER. If the APMSE
1835 attempted to add a group table entry but there was no room in the table for another entry, then the status value
1836 will be TABLE_FULL.

1837 **2.2.4.5.2.3 Effect on Receipt**

1838 On receipt of this primitive, the next higher layer is informed of the status of its request to add a group. The
1839 Status parameter values will be as described above.

1840 **2.2.4.5.3 APSME-REMOVE-GROUP.request**

1841 This primitive allows the next higher layer to request that group membership in a particular group for a par-
1842 ticular endpoint be removed.

1843 **2.2.4.5.3.1 Semantics of the Service Primitive**

1844 The semantics of the service primitive are as follows:

1845 APSME-REMOVE-GROUP.request {
1846 GroupAddress,
1847 Endpoint
1848 }

1849

1850 Table 2.16 specifies the parameters for this primitive.

1851 **Table 2.16 APSME-REMOVE-GROUP.request Parameters**

Name	Type	Valid Range	Description
GroupAddress	16-bit group address	0x0000 - 0xffff	The 16-bit address of the group being removed.
Endpoint	Integer	0x01 - 0xfe	The endpoint to which the given group is being removed.

1852 **2.2.4.5.3.2 When Generated**

1853 This primitive is generated by the next higher layer when it wants to remove membership in a particular
1854 group from an endpoint so that frames addressed to the group will no longer be delivered to that endpoint.

1855 **2.2.4.5.3.3 Effect on Receipt**

1856 If, on receipt of this primitive, the GroupAddress parameter is found to be outside the valid range, then the
1857 APSME will issue the APSME-REMOVE-GROUP.confirm primitive to the next higher layer with a status
1858 value of INVALID_PARAMETER. Similarly, if the Endpoint parameter has a value which is out of range or
1859 else enumerates an endpoint that is not implemented on the current device, the APSME will issue the
1860 APSME-REMOVE-GROUP.confirm primitive with a Status of INVALID_PARAMETER.

1861 After checking the parameters as described above, the APSME will check the group table to see if an entry
1862 exists containing the values given by the GroupAddress and Endpoint parameters. If such an entry already
1863 exists in the table, then that entry will be removed. If the NWK layer is maintaining a group table, then the
1864 APSME ensures that the NWK group table is consistent by issuing the NLME-SET.request primitive, for the
1865 *nwkGroupIDTable* attribute, with the list of group addresses contained in the group table of the APS
1866 sub-layer. Once both tables are consistent, the APSME issues the APSME-REMOVE-GROUP.confirm
1867 primitive to the next higher layer with a status value of SUCCESS. If there is no such entry, the APSME will
1868 issue the APSME-REMOVE-GROUP.confirm primitive to the next higher layer with a status value of
1869 INVALID_GROUP.

1870 **2.2.4.5.4 APSME-REMOVE-GROUP.confirm**

1871 This primitive allows the next higher layer to be informed of the results of its request to remove a group from
1872 an endpoint.

1873 **2.2.4.5.4.1 Semantics of the Service Primitive**

1874 The semantics of the service primitive are as follows:

```
1875           APSME-REMOVE-GROUP.confirm       {  
1876                                                  Status,  
1877                                                  GroupAddress,  
1878                                                  Endpoint  
1879                                                  }
```

1880

1881 Table 2.17 specifies the parameters for this primitive.

1882 **Table 2.17 APSME-REMOVE-GROUP.confirm Parameters**

Name	Type	Valid Range	Description
Status	Enumeration	SUCCESS, INVALID_GROUP or INVALID_PARAMETER	The status of the request to remove a group.
GroupAddress	16-bit group address	0x0000 - 0xffff	The 16-bit address of the group being removed.
Endpoint	Integer	0x01 - 0xfe	The endpoint which is to be removed from the group.

1883 **2.2.4.5.4.2 When Generated**

1884 This primitive is generated by the APSME and issued to the next higher layer in response to an
1885 APMSE-REMOVE-GROUP.request primitive. If the APSME-REMOVE-GROUP.request was successful,
1886 the Status parameter value will be SUCCESS. If the APSME-REMOVE-GROUP.request was not successful
1887 because an entry containing the values given by the GroupAddress and Endpoint parameters did not exist,
1888 then the status value will be INVALID_GROUP. If one of the parameters of the
1889 APMSE-REMOVE-GROUP.request primitive had an invalid value, then the status value will be
1890 INVALID_PARAMETER.

1891 **2.2.4.5.4.3 Effect on Receipt**

1892 On receipt of this primitive, the next higher layer is informed of the status of its request to remove a group.
1893 The Status parameter values will be as described above.

1894 **2.2.4.5.5 APSME-REMOVE-ALL-GROUPS.request**

1895 This primitive is generated by the next higher layer when it wants to remove membership in all groups from
1896 an endpoint, so that no group-addressed frames will be delivered to that endpoint.

1897 **2.2.4.5.5.1 Semantics of the Service Primitive**

1898 The semantics of the service primitive are as follows:

```
1899           APSME-REMOVE-ALL-GROUPS.request {
1900                 Endpoint
1901             }
```

1902

1903 Table 2.18 specifies the parameters for this primitive.

1904 **Table 2.18 APSME-REMOVE-ALL-GROUPS.request Parameters**

Name	Type	Valid Range	Description
Endpoint	Integer	0x01 - 0xfe	The endpoint to which the given group is being removed.

1905 **2.2.4.5.5.2 When Generated**

1906 This primitive is generated by the next higher layer when it wants to remove membership in all groups from
1907 an endpoint so that no group addressed frames will be delivered to that endpoint.

1908 **2.2.4.5.5.3 Effect on Receipt**

1909 If, on receipt of this primitive, the Endpoint parameter has a value which is out of range or else enumerates an
1910 endpoint that is not implemented on the current device the APSME will issue the
1911 APSME-REMOVE-ALL-GROUPS.confirm primitive with a Status of INVALID_PARAMETER.

1912 After checking the Endpoint parameter as described above, the APSME will remove all entries related to this
1913 endpoint from the group table. The APSME ensures that the corresponding NWK group table is consistent by
1914 issuing the NLME-SET.request primitive, for the *nwkGroupIDTable* attribute, with the list of group ad-
1915 dresses contained in the group table of the APS sub-layer. Once both tables are consistent, the APSME issues
1916 the APSME-REMOVE-ALL-GROUPS.confirm primitive to the next higher layer with a status value of
1917 SUCCESS.

1918 **2.2.4.5.6 APSME-REMOVE-ALL-GROUPS.confirm**

1919 This primitive allows the next higher layer to be informed of the results of its request to remove all groups
1920 from an endpoint.

1921 **2.2.4.5.6.1 Semantics of the Service Primitive**

1922 The semantics of the service primitive are as follows:

```
1923 APSME-REMOVE-ALL-GROUPS.confirm {
1924     Status,
1925     Endpoint
1926 }
```

1927

1928 Table 2.19 specifies the parameters for this primitive.

1929 **Table 2.19 APSME-REMOVE-ALL-GROUPS.confirm Parameters**

Name	Type	Valid Range	Description
Status	Enumeration	SUCCESS or INVALID_PARAMETER	The status of the request to remove all groups.
Endpoint	Integer	0x01 - 0xfe	The endpoint which is to be removed from all groups.

1930 **2.2.4.5.6.2 When Generated**

1931 This primitive is generated by the APSME and issued to the next higher layer in response to an
1932 APSME-REMOVE-ALL-GROUPS.request primitive. If the APSME-REMOVE-ALL-GROUPS.request
1933 was successful, then the Status parameter value will be SUCCESS. If the Endpoint parameter of the
1934 APSME-REMOVE-ALL-GROUPS.request primitive had an invalid value, then the status value will be
1935 INVALID_PARAMETER.

1936 **2.2.4.5.6.3 Effect on Receipt**

1937 On receipt of this primitive, the next higher layer is informed of the status of its request to remove all groups
1938 from an endpoint. The Status parameter values will be as described above.

1939 2.2.5 Frame Formats

1940 This section specifies the format of the APS frame (APDU). Each APS frame consists of the following
1941 basic components:

- 1942 • An APS header, which comprises frame control and addressing information.
1943 • An APS payload, of variable length, which contains information specific to the frame type.

1944 The frames in the APS sub-layer are described as a sequence of fields in a specific order. All frame formats
1945 in this section are depicted in the order in which they are transmitted by the NWK layer, from left to right,
1946 where the leftmost bit is transmitted first in time. Bits within each field are numbered from 0 (leftmost and
1947 least significant) to k-1 (rightmost and most significant), where the length of the field is k bits. Fields that
1948 are longer than a single octet are sent to the NWK layer in order from the octet containing the low-
1949 est-numbered bits to the octet containing the highest-numbered bits.

1950 On transmission, all fields marked as reserved shall be set to zero. On reception, all fields marked as re-
1951 served in this version of the specification shall be checked for being equal to zero. If such a reserved field is
1952 not equal to zero, no further processing shall be applied to the frame and the frame shall be discarded.

1953 2.2.5.1 General APDU Frame Format

1954 The APS frame format is composed of an APS header and an APS payload. The fields of the APS header
1955 appear in a fixed order, however, the addressing fields may not be included in all frames. The general APS
1956 frame shall be formatted as illustrated in Figure 2.2.

1957 Figure 2.2 General APS Frame Format

Octets: 1	0/1	0/2	0/2	0/2	0/1	1	0/ Variable	Variable
Frame control	Destination endpoint	Group address	Cluster identifier	Profile identifier	Source endpoint	APS counter	Extended header	Frame payload
	Addressing fields							
APS header								APS pay- load

1958 2.2.5.1.1 Frame Control Field

1959 The frame control field is 8-bits in length and contains information defining the frame type, addressing
1960 fields, and other control flags. The frame control field shall be formatted as illustrated in Figure 2.3.

1961 Figure 2.3 Format of the Frame Control Field

Bits: 0-1	2-3	4	5	6	7
Frame type	Delivery mode	Ack. format	Security	Ack. request	Extended header present

1962 2.2.5.1.1.1 Frame Type Sub-Field

1963 The frame type sub-field is two bits in length and shall be set to one of the non-reserved values listed in
1964 Table 2.20.

1965

Table 2.20 Values of the Frame Type Sub-Field

Frame Type Value $b_1\ b_0$	Frame Type Name
00	Data
01	Command
10	Acknowledgement
11	Inter-PAN APS

1966

2.2.5.1.1.2 Delivery Mode Sub-Field

1967
1968

The delivery mode sub-field is two bits in length and shall be set to one of the non-reserved values from Table 2.21.

1969

Table 2.21 Values of the Delivery Mode Sub-Field

Delivery Mode Value $b_1\ b_0$	Delivery Mode Name
00	Normal unicast delivery
01	Reserved
10	Broadcast
11	Group addressing

1970

1971

If the value is 0b00, the frame will be delivered to a given endpoint on the receiving device.

1972
1973
1974
1975

If the value is 0b10, the message is a broadcast. In this case, the message will go to all devices defined for the selected broadcast address in use as defined in section 3.6.5. The destination endpoint field shall be set to a value between 0x01-0xfe (for broadcasts to specific endpoints) or to 0xff (for broadcasts to all active endpoints).

1976
1977
1978
1979
1980

If the value is 0b11, then group addressing is in use and that frame will only be delivered to device endpoints that express group membership in the group identified by the group address field in the APS header. Note that other endpoints on the source device may be members of the group addressed by the outgoing frame. The frame shall be delivered to any member of the group, including other endpoints on the source device that are members of the specified group.

1981
1982
1983

Devices where nwkUseMulticast is set to TRUE, shall never set the delivery mode of an outgoing frame to 0b11. In this case, the delivery mode of the outgoing frame shall be set to 0b10 (broadcast) and the frame shall be sent using an NLDE-DATA.request with the destination address mode set to group addressing.

1984	2.2.5.1.1.3	Ack Format Field
1985		This bit indicates if the destination endpoint, cluster identifier, profile identifier and source endpoint fields shall be present in the acknowledgement frame. This is set to 0 for data frame acknowledgement and 1 for APS command frame acknowledgement.
1988	2.2.5.1.1.4	Security Sub-Field
1989		The Security Services Provider (see Chapter 4) manages the security sub-field.
1990	2.2.5.1.1.5	Acknowledgement Request Sub-Field
1991		The acknowledgement request sub-field is one bit in length and specifies whether the current transmission requires an acknowledgement frame to be sent to the originator on receipt of the frame. If this sub-field is set to 1, the recipient shall construct and send an acknowledgement frame back to the originator after determining that the frame is valid. If this sub-field is set to 0, the recipient shall not send an acknowledgement frame back to the originator.
1996		This sub-field shall be set to 0 for all frames that are broadcast or multicast.
1997	2.2.5.1.1.6	Extended Header Present
1998		The extended header present sub-field is one bit in length and specifies whether the extended header shall be included in the frame. If this sub-field is set to 1, then the extended header shall be included in the frame. Otherwise, it shall not be included in the frame.
2001	2.2.5.1.2	Destination Endpoint Field
2002		The destination endpoint field is 8-bits in length and specifies the endpoint of the final recipient of the frame. This frame shall be included in the frame only if the delivery mode subfield is set to 0b00 (normal unicast delivery), or 0b10 (broadcast delivery). In the case of broadcast delivery, the frame shall be delivered to the destination endpoint specified within the range 0x01-0xfe or to all active endpoints if specified as 0xff.
2007		A destination endpoint value of 0x00 addresses the frame to the ZigBee device object (ZDO), resident in each device. A destination endpoint value of 0x01-0xfe addresses the frame to an application operating on that endpoint. A destination endpoint value of 0xff addresses the frame to all active endpoints except endpoint 0x00.
2011	2.2.5.1.3	Group Address Field
2012		The group address field is 16 bits in length and will only be present if the delivery mode sub-field of the frame control has a value of 0b11. In this case, the destination endpoint shall not be present. If the APS header of a frame contains a group address field, the frame will be delivered to all endpoints for which the group table in the device contains an association between that endpoint and the group identified by the contents of the group address field.
2017		Devices where <i>nwkUseMulticast</i> is set to TRUE shall never set the group address field of an outgoing frame.
2019	2.2.5.1.4	Cluster Identifier Field
2020		The cluster identifier field is 16 bits in length and specifies the identifier of the cluster to which the frame relates and which shall be made available for filtering and interpretation of messages at each device that takes delivery of the frame. This field shall be present only for data or acknowledgement frames.
2023	2.2.5.1.5	Profile Identifier Field
2024		The profile identifier is two octets in length and specifies the ZigBee profile identifier for which the frame is intended and shall be used during the filtering of messages at each device that takes delivery of the frame. This field shall be present only for data or acknowledgement frames.

2027 **2.2.5.1.6 Source Endpoint Field**

2028 The source endpoint field is eight-bits in length and specifies the endpoint of the initial originator of the
2029 frame. A source endpoint value of 0x00 indicates that the frame originated from the ZigBee device object
2030 (ZDO) resident in each device. A source endpoint value of 0x01-0xfe indicates that the frame originated
2031 from an application operating on that endpoint.

2032 **2.2.5.1.7 APS Counter**

2033 This field is eight bits in length and is used as described in section 2.2.8.4.2 to prevent the reception of du-
2034 plicate frames. This value shall be incremented by one for each new transmission.

2035 **2.2.5.1.8 Extended Header Sub-Frame**

2036 The extended header sub-frame contains further sub-fields and shall be formatted as illustrated in Figure
2037 2.4.

2038 **Figure 2.4 Format of the Extended Header Sub-Frame**

Octets: 1	0/1	0/1
Extended frame control	Block number	ACK bitfield

2039 **2.2.5.1.8.1 Extended Frame Control Field**

2040 The extended frame control field is eight-bits in length and contains information defining the use of frag-
2041 mentation. The extended frame control field shall be formatted as illustrated in Figure 2.5.

2042 **Figure 2.5 Format of the Extended Frame Control Field**

Bits: 0-1	2-7
Fragmentation	Reserved

2043

2044 The fragmentation sub-field is two bits in length and shall be set to one of the non-reserved values listed in
2045 Table 2.22.

2046 **Table 2.22 Values of the Fragmentation Sub-Field**

Fragmentation Value $b_1\ b_0$	Description
00	Transmission is not fragmented.
01	Frame is first fragment of a fragmented transmission.
10	Frame is part of a fragmented transmission but not the first part.
11	Reserved

2047 **2.2.5.1.8.2 Block Number**

2048 The block number field is one octet in length and is used for fragmentation control as follows: If the fragmentation sub-field is set to indicate that the transmission is not fragmented then the block number field shall not be included in the sub-frame. If the fragmentation sub-field is set to 01, then the block number field shall be included in the sub-frame and shall indicate the number of blocks in the fragmented transmission. If the fragmentation sub-field is set to 10, then the block number field shall be included in the sub-frame and shall indicate which block number of the transmission the current frame represents, taking the value 0x01 for the second fragment, 0x02 for the third, etc.

2055 **2.2.5.1.8.3 Ack Bitfield**

2056 The ack bitfield field is one octet in length and is used in an APS acknowledgement as described in section 2057 2.2.8.4.5.2 to indicate which blocks of a fragmented ASDU have been successfully received. This field is 2058 only present if the frame type sub-field indicates an acknowledgement and the fragmentation sub-field indicates 2059 a fragmented transmission.

2060 **2.2.5.1.9 Frame Payload Field**

2061 The frame payload field has a variable length and contains information specific to individual frame types.

2062 **2.2.5.2 Format of Individual Frame Types**

2063 There are three defined frame types: data, APS command, and acknowledgement. Each of these frame 2064 types is discussed in the following sections.

2065 **2.2.5.2.1 Data Frame Format**

2066 The data frame shall be formatted as illustrated in Figure 2.6.

2067 **Figure 2.6 Data Frame Format**

Octets: 1	0/1	0/2	2	2	1	1	0/ Variable	Variable	
Frame control	Destination endpoint	Group address	Cluster identifier	Profile Identifier	Source endpoint	APS counter	Extended header	Frame payload	
	Addressing fields								
APS header							APS pay- load		

2068

2069 The order of the fields of the data frame shall conform to the order of the general APS frame as illustrated 2070 in Figure 2.2.

2071 **2.2.5.2.1.1 Data Frame APS Header Fields**

2072 The APS header field for a data frame shall contain the frame control, cluster identifier, profile identifier, 2073 source endpoint and APS counter fields. The destination endpoint, group address and extended header 2074 fields shall be included in a data frame according to the values of the delivery mode and extended header 2075 present sub-fields of the frame control field.

2076 In the frame control field, the frame type sub-field shall contain the value that indicates a data frame, as 2077 shown in Table 2.20. All other sub-fields shall be set appropriately according to the intended use of the data 2078 frame.

2079 **2.2.5.2.1.2 Data Payload Field**

2080 For an outgoing data frame, the data payload field shall contain part or all of the sequence of octets that the
2081 next higher layer has requested the APS data service to transmit. For an incoming data frame, the data pay-
2082 load field shall contain all or part of the sequence of octets that has been received by the APS data service
2083 and that is to be delivered to the next higher layer.

2084 **2.2.5.2.2 APS Command Frame Format**

2085 The APS command frame shall be formatted as illustrated in Figure 2.7.

2086 **Figure 2.7 APS Command Frame Format**

Octets: 1	1	1	Variable
Frame control	APS counter	APS command identifier	APS command payload
APS header	APS payload		

2087 The order of the fields of the APS command frame shall conform to the order of the general APS frame as
2088 illustrated in Figure 2.2.

2089 **2.2.5.2.2.1 APS Command Frame APS Header Fields**

2090 The APS header field for an APS command frame shall contain the frame control and APS counter fields.
2091 In this version of the specification, the APS command frame shall not be fragmented and the extended
2092 header field shall not be present.

2093 In the frame control field, the frame type sub-field shall contain the value that indicates an APS command
2094 frame, as shown in Table 2.20. The APS Command Payload shall be set appropriately according to the in-
2095 tended use of the APS command frame.

2096 **2.2.5.2.2.2 APS Command Identifier Field**

2097 The APS command identifier field identifies the APS command being used.

2098 **2.2.5.2.2.3 APS Command Payload Field**

2099 The APS command payload field of an APS command frame shall contain the APS command itself.

2100 **2.2.5.2.3 Acknowledgement Frame Format**

2101 The acknowledgement frame shall be formatted as illustrated in Figure 2.8.

2102 **Figure 2.8 Acknowledgement Frame Format**

Octets: 1	0/1	0/2	0/2	0/1	1	0/Variable
Frame control	Destination endpoint	Cluster identifier	Profile identifier	Source endpoint	APS counter	Extended header
APS header						

2103 The order of the fields of the acknowledgement frame shall conform to the order of the general APS frame
2104 as illustrated in Figure 2.2.

2105 **2.2.5.2.3.1 Acknowledgement Frame APS Header Fields**

2106 If the ack format field is not set in the frame control field, the destination endpoint, cluster identifier, profile
2107 identifier and source endpoint shall be present. This is not set for data frame acknowledgement. The ex-
2108 tended header field shall be included in a data frame according to the value of the extended header present
2109 sub-field of the frame control field.

2110 In the frame control field, the frame type sub-field shall contain the value that indicates an acknowledg-
2111 ement frame, as shown in Table 2.20. The extended header present sub-field shall contain the same value as
2112 in the frame to which this frame is an acknowledgement. All other sub-fields shall be set appropriately ac-
2113 cording to the intended use of the acknowledgement frame.

2114 If the ack format field is set in the frame control field, the frame is an APS command frame acknowledg-
2115 ement and the destination endpoint, cluster identifier, profile identifier and source endpoint fields shall not
2116 be included. Alternatively, if an APS data frame is being acknowledged, the source endpoint field shall re-
2117 flect the value in the destination endpoint field of the frame that is being acknowledged. Similarly, the des-
2118 tination endpoint field shall reflect the value in the source endpoint field of the frame that is being
2119 acknowledged. And the Cluster identifier and Profile identifier fields shall contain the same values as in the
2120 frame to which this frame is an acknowledgement.

2121 The APS counter field shall contain the same value as the frame to which this frame is an acknowledgment.

2122 Where the extended header is present, the fragmentation sub-field of the extended frame control field shall
2123 contain the same value as in the frame to which this frame is an acknowledgement. If fragmentation is in
2124 use for this frame, then the block number and ack bitfield fields shall be present. Where present, the block
2125 number field shall contain block number to which this frame is an acknowledgement. If fragmentation is in
2126 use, the acknowledgement frames shall be issued according to section 2.2.8.4.5.2 and not for each received
2127 frame unless the transmission window size is set to request acknowledgement of each frame.

2128 **2.2.6 Command Frames**

2129 This specification defines no command frames. Refer to section 4.4.9 for a thorough description of the APS
2130 command frames and primitives related to security.

2131 **2.2.7 Constants and PIB Attributes**

2132 **2.2.7.1 APS Constants**

2133 The constants that define the characteristics of the APS sub-layer are presented in Table 2.23.

2134 **Table 2.23 APS Sub-Layer Constants**

Constant	Description	Value
apscMaxDescriptorSize	The maximum number of octets contained in a non-complex descriptor.	64
apscMaxFrameRetries	The maximum number of retries allowed after a transmission failure.	3

Constant	Description	Value
apscAckWaitDuration	The maximum number of seconds to wait for an acknowledgement to a transmitted frame.	0.05 * (2*nwkcMaxDepth) + (security encrypt/decrypt delay), where the (security encrypt/decrypt delay) = 0.1 (assume 0.05 per encrypt or decrypt cycle)
apscMinDuplicateRejectionTableSize	The minimum required size of the APS duplicate rejection table.	1
apscMinHeaderOverhead	The minimum number of octets added by the APS sub-layer to an ASDU.	0x0C
apsParentAnnounceBaseTimer	The base amount of delay before each broadcast parent announce is sent.	10
apsParentAnnounceJitterMax	The max amount of jitter that is added to the apsParentAnnounceBaseTimer before each broadcast parent announce is sent.	10

2.2.7.2 APS Information Base

The APS information base comprises the attributes required to manage the APS layer of a device. The attributes of the AIB are listed in Table 2.24. The security-related AIB attributes are described in section 4.4.10.

Table 2.24 APS IB Attributes

Attribute	Identifier	Type	Range	Description	Default
apsBindingTable	0xc1	Set	Variable	The current set of binding table entries in the device (see section 2.2.8.2.1).	Null set
apsDesignated-Coordinator	0xc2	Boolean	TRUE or FALSE	TRUE if the device should become the ZigBee Coordinator on startup, FALSE if otherwise.	FALSE
apsChannelMask	0xc3	IEEE 802.15.4 channel mask	Any legal mask for the PHY	The mask of allowable channels for this device to use for network operations.	All channels

Attribute	Identifier	Type	Range	Description	Default
apsUseExtended-PANID	0xc4	64-bit extended address	0x0000000000000000 to 0xfffffffffffffe	The 64-bit address of a network to form or to join.	0x00000000 00000000
apsGroupTable	0x0c5	Set	Variable	The current set of group table entries (see Table 2.25).	Null set
apsNonmember Radius	0xc6	Integer	0x00-0x07	The value to be used for the NonmemberRadius parameter when using NWK layer multicast.	2
apsUseInsecure-Join	0xc8	Boolean	TRUE or FALSE	A flag controlling the use of insecure join at startup.	FALSE
apsInter-frameDelay	0xc9	Integer	0x00 to 0xff (may be restricted by application profile)	Fragmentation parameter—the standard delay, in milliseconds, between sending two blocks of a fragmented transmission (see section 2.2.8.4.5).	Set by application profile
apsLastChannel Energy	0xca	Integer	0x00 - 0xff	The energy measurement for the channel energy scan performed on the previous channel just before a channel change (in accordance with [B1]).	Null set
apsLastChannel FailureRate	xcb	Integer	0-100 (decimal)	The latest percentage of transmission network transmission failures for the previous channel just before a channel change (in percentage of failed transmissions to the total number of transmissions attempted)	Null set

Attribute	Identifier	Type	Range	Description	Default
apsChannelTimer	0xcc	Integer	1-24 (decimal)	A countdown timer (in hours) indicating the time to the next permitted frequency agility channel change. A value of NULL indicates the channel has not been changed previously.	Null set
apsMaxWindowSize	0xcd	See Table 2.26	Variable	A table with the active endpoints and their respective <i>apsMaxWindowSize</i> where fragmentation is used (active endpoints not supporting fragmentations shall be omitted from the list).	Null set
ap-sParentAnnounceTimer	0xce	Intege r	0 to ap-sParentAnnounceBaseTimer + ap-sParentAnnounceJitterMax	The value of the current countdown timer before the next Parent_annce is sent.	0

2140

2141

Table 2.25 Group Table Entry Format

Group ID	Endpoint List
16-bit group address	List of endpoints on this device which are members of the group.

2142

2143

Table 2.26 *apsMaxWindowSize* by Endpoint Number

Endpoint Number	apsMaxWindowSize for the Endpoint Number
0x01 - 0xff	Value of 1-8

2144

2145 2.2.8 Functional Description

2146 2.2.8.1 Persistent Data

2147 The APS is required to maintain a minimum set of data in persistent memory. This data set shall persist
2148 over power fail, device reset, or other processing events. The following data shall be maintained in persis-
2149 tent memory within APS:

- 2150 • *apsBindingTable* (if supported on the device)
- 2151 • *apsDesignatedCoordinator* (if supported on the device)
- 2152 • *apsChannelMask*
- 2153 • *apsUseExtendedPANID*
- 2154 • *apsUseInsecureJoin*
- 2155 • *apsGroupTable* (if supported on the device)
- 2156 • Binding Table Cache (if the device is designated as a primary or backup binding table cache, see sec-
2157 tion 2.4.2.4)
- 2158 • Discovery Cache (if the device is designated as a primary discovery cache, see section 2.4.2.1)
- 2159 • Node Descriptor, Power Descriptor plus the Simple Descriptor(s) for each active endpoint on the de-
2160 vice
- 2161 • Network manager address

2162 The method by which these data are made to persist is beyond the scope of this specification.

2163 2.2.8.2 Binding

2164 The APS may maintain a binding table, which allows ZigBee devices to establish a designated destination
2165 for frames from a given source endpoint and with a given cluster ID. Each designated destination shall rep-
2166 resent either a specific endpoint on a specific device, or a group address.

2167 2.2.8.2.1 Binding Table Implementation

2168 A device designated as containing a binding table shall be able to support a binding table of implemen-
2169 tation-specific length. The binding table shall implement the following mapping:

$$2170 (a_s, e_s, c_s) = \{(a_{d1}|, e_{d1}|), (a_{d2}|, e_{d2}|) \dots (a_{dn}|, e_{dn}|)\}$$

2171 Where:

a_s	= the address of the device as the source of the binding link
e_s	= the endpoint identifier of the device as the source of the binding link
c_s	= the cluster identifier used in the binding link
a_{di}	= the i^{th} destination address or destination group address associated with the binding link
e_{di}	= the i^{th} optional destination endpoint identifier associated with the binding link Note that e_{di} will only be present when a_{di} is a device address.

2172 **2.2.8.2.2 Binding**

2173 The APSME-BIND.request or APSME-UNBIND.request primitives initiate the procedure for creating or
2174 removing a binding link. Only a device supporting a binding table cache, or a device that wishes to store
2175 source bindings, shall initiate this procedure. If this procedure is initiated by another type of device, then
2176 the APSME shall issue the APSME-BIND.confirm or APSME-UNBIND.confirm primitive with the Status
2177 parameter set to ILLEGAL_REQUEST.

2178 When this procedure is initiated, the APSME shall first extract the address and endpoint for both the source
2179 and destination of the binding link. If the DstAddrMode parameter has a value of 0x01, indicating group
2180 addressing, then only the source address is treated in the way just described. The 16-bit group address is
2181 used directly as a destination address and, in this case, no destination endpoint is specified. With this in-
2182 formation, the APSME shall either create a new entry or remove the corresponding entry from its binding
2183 table, depending on whether the bind or unbind procedure, respectively, was initiated.

2184 If a bind operation was requested, the APSME shall create a new entry in the binding table. The device
2185 shall only create a new entry in the binding table if it has the capacity to do so. If the binding table does not
2186 have capacity, then the APSME shall issue the APSME-BIND.confirm primitive with the Status parameter
2187 set to TABLE_FULL.

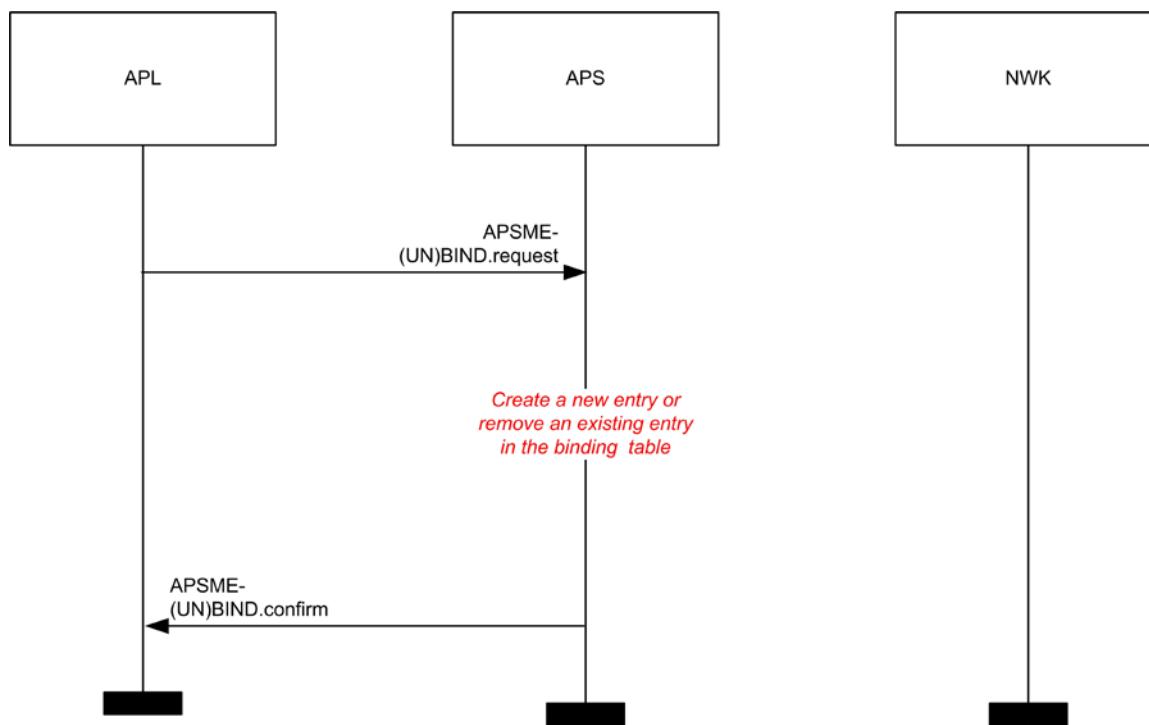
2188 If an unbind operation was requested, the APSME shall search the binding table for an existing entry that
2189 matches the information contained in the initiation request. If an entry is not found, the APSME shall ter-
2190 minate the procedure and notify the NHLE of the invalid binding. This is achieved by issuing the
2191 APSME-UNBIND.confirm primitive with the Status parameter set to INVALID_BINDING. If an entry is
2192 found, the APSME shall remove the entry in the binding table.

2193 If the binding link is successfully created or removed, the APSME shall notify the NHLE of the results of
2194 the binding attempt and the success of the procedure. This is achieved by issuing the
2195 APSME-BIND.confirm or APSME-UNBIND.confirm primitive, respectively, with the binding results and
2196 the Status parameter set to SUCCESS.

2197 The procedure for a successful binding is illustrated in the MSC shown in Figure 2.9.

2198

Figure 2.9. Binding on a Device Supporting a Binding Table



2199

2200

2.2.8.3 Group Addressing

2202 The APS sub-layer shall maintain a group table, which allows endpoints to be associated with groups and
2203 allows group-addressed frames to be delivered selectively to those endpoints that are associated in the table
2204 with a particular group.

2205 The list of group addresses in the APS sub-layer group table shall be kept consistent with the list of group
2206 IDs in the NWK layer group table, stored in the *nwkGroupIDTable* attribute.

2.2.8.3.1 The Group Table

2208 For purposes of this discussion, the group table shall be viewed as a set of associations between groups and
2209 endpoints as follows:

$$\{(g_1 - ep_{11}, ep_{12}...ep_{1n}), (g_2 - ep_{21}, ep_{22}...ep_{2m})... (g_i - ep_{i1}, ep_{i2}...ep_{ik})\}$$

2211 where:

$$g_i = \text{the } i^{\text{th}} \text{ group represented in the table}$$
$$ep_{ij} = \text{the } j^{\text{th}} \text{ endpoint associated with the } i^{\text{th}} \text{ group}$$

2212 Implementers of this specification are free to implement the group table in any manner that is convenient
2213 and efficient, as long as it represents the associations just described.

2.2.8.4 Transmission, Reception, and Acknowledgement

2214 This section describes the fundamental procedures for transmission, reception, and acknowledgement.

2216 **2.2.8.4.1 Transmission**

2217 Only those devices that are currently part of a network shall send frames from the APS sub-layer. If any
2218 other device receives a request to transmit a frame, it shall discard the frame and notify the instigating layer
2219 of the error. An APSDE-DATA.confirm primitive with a status of CHANNEL_ACCESS_FAILURE indicates
2220 that the attempt at transmission of the frame was unsuccessful due to the channel being busy.

2221 All frames handled by or generated within the APS sub-layer shall be constructed according to the general
2222 frame format specified in section 2.2.5.1 and transmitted using the NWK layer data service.

2223 Transmissions employing delivery modes 0b00 (Normal Unicast) and 0b10 (Broadcast) shall include both
2224 the source endpoint and destination endpoint fields. Group addressed transmissions, having a delivery
2225 mode sub-field value of 0b11 shall contain a source endpoint field and group address field, but no destination
2226 endpoint field. Note that other endpoints on the source device are legal group members and possible
2227 destinations for group-addressed frames.

2228 For all devices where the transmission is due to a binding table entry stored on the source device, the
2229 APSDE of the source device shall determine whether the binding table entry contains a unicast destination
2230 device address or a destination group address. In the case where a binding table entry contains a unicast
2231 destination device address and this destination device address is that of the source device itself, the APSDE
2232 shall issue an APSDE-DATA.indication primitive to the next higher layer and shall not transmit a frame.
2233 Otherwise, the APSDE shall transmit the frame to the 16-bit NWK address corresponding to the destination
2234 address indicated by the binding table entry, and the delivery mode sub-field of the frame control field shall
2235 be set to 0b00. In the case where the binding table entry contains a destination group address and nwkUse
2236 Multicast is FALSE, the delivery mode sub-field of the frame control field shall have a value of 0b11, the
2237 destination group address shall be placed in the APS header, and the destination endpoint shall be omitted.
2238 The frame shall then be broadcast using the NLDE-DATA.request primitive and employing a broadcast
2239 address of 0xffffd. In the case where the binding table entry contains a destination group address and
2240 nwkUseMulticast is TRUE, the delivery mode sub-field of the frame control field shall have a value of
2241 0b10 and the destination endpoint shall have a value of 0xff. The frame shall then be multicast using the
2242 NLDE-DATA.request primitive and employing the group address supplied in the binding table entry.

2243 If security is required, the frame shall be processed as described in section 4.4.

2244 If fragmentation is required, and is permitted for this frame, then the frame shall be processed as described
2245 in section 2.2.8.4.5.

2246 When the frame is constructed and ready for transmission, it shall be passed to the NWK data service with
2247 suitable destination and source addresses. In addition, the APS layer shall ensure that route discovery is
2248 enabled at the network layer. An APDU transmission is initiated by issuing the NLDE-DATA.request
2249 primitive to the NWK layer and the results of the transmission returned via the NLDE-DATA.confirm
2250 primitive.

2251 **2.2.8.4.2 Reception and Rejection**

2252 The APS sub-layer shall be able to filter frames arriving via the NWK layer data service and only present
2253 the frames that are of interest to the NHLE.

2254 If the APSDE receives a secured frame, it shall process the frame as described in section 4.4 to remove the
2255 security.

2256 If the APSDE receives a frame containing the destination endpoint field, then the APSDE shall pass it directly
2257 to the NHLE at the destination endpoint supplied, unless it is part of an incomplete fragmented
2258 transmission or it is determined to have been a duplicate of a frame that has been passed up previously.
2259 Subject to the same incomplete fragmented transmission and duplicate frame detection, if the destination
2260 endpoint is set to the broadcast endpoint (0xff) and the DstAddrMode parameter of the received
2261 NLDE-DATA.indication primitive was not 0x01, then the APSDE shall also present the frame to all
2262 non-reserved endpoints (0x01-0xfe) supported by the NHLE.

If the APSDE of a device receives a transmission with the delivery mode sub-field of the frame control field set to 0b11, indicating group addressing, it shall deliver the frame to each endpoint on the device that is associated in the group table with the 16-bit group address found in the group address field of the APS header. Similarly, if the APSDE of a device receives a NLDE-DATA.indication primitive where the DstAddrMode parameter has a value of 0x01, also indicating group addressing, it shall deliver the frame to each endpoint on the device that is associated in the group table with the 16-bit group address given as the value of the DstAddr parameter. In either case, it shall search the group table and, for each endpoint associated with the given group address, it shall issue the NLDE-DATA.indication primitive to the next higher layer with a value of the DstEndpoint parameter equal to the number of the associated endpoint. All other parameters of the NLDE-DATA.indication primitive shall remain the same for all instances of the primitive issued.

The APSDE shall maintain a duplicate rejection table to include at least source address, APS counter, and timing information, such that frames transmitted according to this specification and received more than once are identified as duplicates and only delivered to the NHLE once. The size of this table shall be at least *apscMinDuplicateRejectionTableSize*.

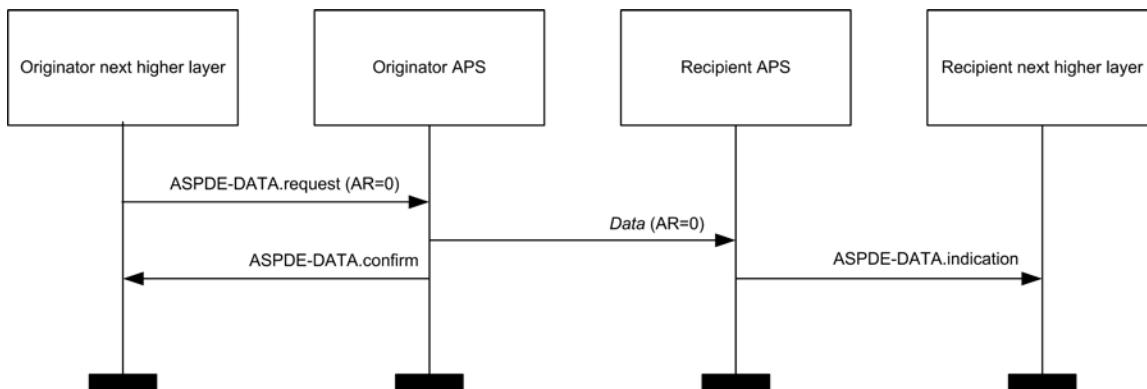
2.2.8.4.3 Use of Acknowledgements

A data or APS command frame shall be sent with its acknowledgement request sub-field set appropriately for the frame. An acknowledgement frame shall always be sent with the acknowledgement request sub-field set to 0. Similarly, any frame that is broadcast or multicast shall be sent with its acknowledgement request sub-field set to 0.

2.2.8.4.3.1 No Acknowledgement

A frame that is received by its intended recipient with its acknowledgement request (AR) sub-field set to 0 shall not be acknowledged. The originating device shall assume that the transmission of the frame was successful. Figure 2.10 shows the scenario for transmitting a single frame of data from an originator to a recipient without requiring an acknowledgement. In this case, the originator transmits the data frame with the AR sub-field equal to 0.

Figure 2.10 Successful Data Transmission Without an Acknowledgement



2.2.8.4.3.2 Acknowledgement

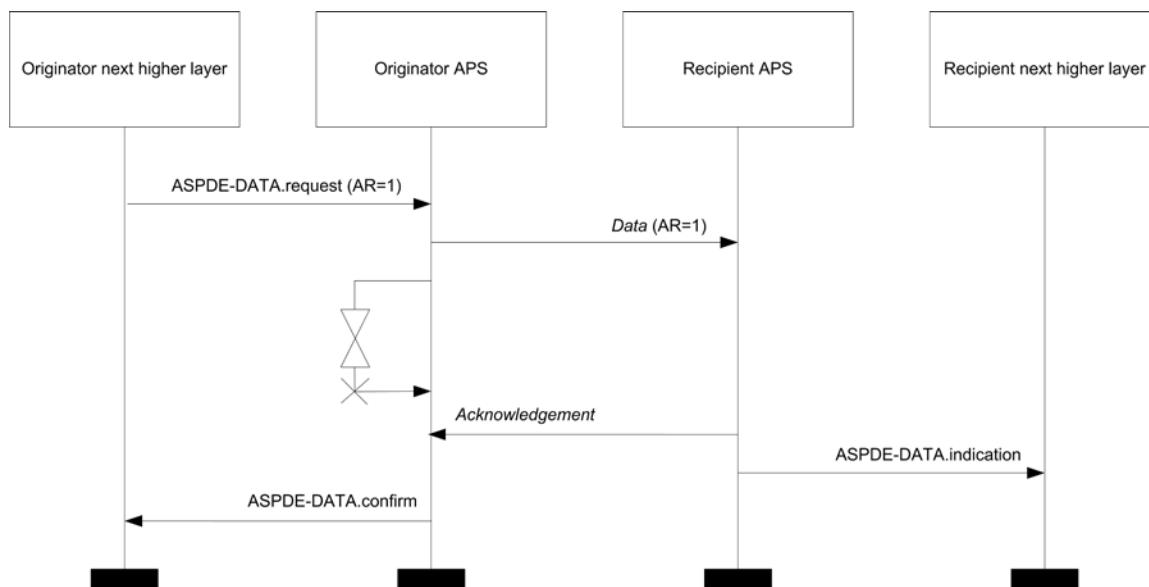
A frame that is received by its intended recipient with its acknowledgement request (AR) sub-field set to 1 shall be acknowledged. If the intended recipient correctly receives the frame, it shall generate and send an acknowledgement frame to the originator of the frame that is being acknowledged.

The transmission of an acknowledgement frame shall commence when the APS sub-layer determines that the frame is valid.

Figure 2.11 shows the scenario for transmitting a single frame of data from an originator to a recipient with an acknowledgement. In this case, the originator indicates to the recipient that it requires an acknowledgement by transmitting the data frame with the AR sub-field set to 1.

2300

Figure 2.11 Successful Data Transmission with an Acknowledgement



2301

2.2.8.4.4 Retransmissions

2303 A device that sends a frame with its acknowledgement request sub-field set to 0 shall assume that the
2304 transmission was successfully received and shall hence not perform the retransmission procedure.

2305 A device that sends a frame with its acknowledgement request sub-field set to 1 shall wait for a maximum
2306 of *apscAckWaitDuration* seconds for the corresponding acknowledgement frame to be received.

2307 If an acknowledgement frame is received within *apscAckWaitDuration* seconds, containing the same cluster
2308 identifier and APS counter as the original frame and has a source endpoint equal to the destination end-
2309 point to which the original frame was transmitted, the transmission shall be considered successful and no
2310 further action shall be taken by the device. If an acknowledgement is not received within *apscAck-
2311 WaitDuration* seconds, or an acknowledgement is received within *apscAckWaitDuration* seconds but con-
2312 tains an unexpected cluster identifier or APS counter or has a source endpoint that is not equal to the desti-
2313 nation endpoint to which the original frame was transmitted, the device shall conclude that the single
2314 transmission attempt has failed.

2315 If a single transmission attempt has failed, the device shall repeat the process of transmitting the frame and
2316 waiting for the acknowledgement, up to a maximum of *apscMaxFrameRetries* times. If an acknowledg-
2317 ment is still not received after *apscMaxFrameRetries* retransmissions, the APS sub-layer shall assume the
2318 transmission has failed and notify the next higher layer of the failure.

2319 Retransmissions of a secured frame shall use a frame counter greater than the original frame.

2.2.8.4.5 Fragmented Transmissions

2321 Where an ASDU is too large to be transmitted within a single MAC data frame, an acknowledged unicast
2322 transmission was requested, and fragmentation is permitted for this frame, the ASDU shall be fragmented
2323 into a number of smaller byte strings, here referred to as “blocks.” Each block is transmitted in a separate
2324 frame.

2325 A “transmission window” is used to arrange an orderly transaction. The window size is set by the stack
2326 profile, and may be set as high as eight blocks. The protocol below arranges that all blocks in a transmis-
2327 sion window must be received and acknowledged before the window can move on. An acknowledgement is
2328 sent when all blocks in the transmission window have been successfully received or, according to the pro-
2329 tocol below, to request retransmission of one or more unreceived blocks.

2330 Transactions not using APS acknowledgements may not be fragmented. Multicast and broadcast transmis-
2331 sions are not permitted to use fragmentation.

2332 **2.2.8.4.5.1 Transmission**

2333 All blocks in a fragmented transmission shall have the same APS Counter value. The extended header
2334 sub-frame shall be included in the frame. The fragmentation sub-field of the extended frame control field
2335 shall be set to 0b01 for the first block and 0b10 for all subsequent blocks of the fragmented transmission.
2336 The block number field shall indicate the total number of blocks in the transmission in the first block, shall
2337 take the value 0x01 in the second block, and thereafter shall be incremented for each subsequent block.

2338 A transmission window shall be maintained, initially covering blocks 0 to (*apscMaxWindowSize*-1), or the
2339 total number of blocks if this is less.

2340 If security is required, then each frame shall be processed independently, as described in clause 4. Following
2341 transmission of each block, the APS shall start a timer. If there are more unacknowledged blocks to
2342 send in the current transmission window, then after a delay of *apsInterframeDelay* milliseconds the next
2343 block shall be passed to the NWK data service. Otherwise, the timer shall be set for *apscAckWaitDuration*
2344 seconds.

2345 A retryCounter parameter shall be maintained and is reset to zero for each new transaction. If an
2346 *apscAckWaitDuration* timer expires, then the block with the lowest unacknowledged block number shall be
2347 passed to the NWK data service again, and the retryCounter parameter shall be incremented. If the re-
2348 tryCounter parameter reaches the value *apscMaxFrameRetries*, the transaction shall be deemed to have
2349 failed, and an APSDE-DATA.confirm primitive returned to the NHLE with a status value of NO_ACK.

2350 On receipt of an acknowledgement frame with matching values in the APS counter, block number, and ad-
2351 dressing fields, outgoing blocks are acknowledged as described in the section below. If at least one previ-
2352 ously unacknowledged block is acknowledged, then the timer shall be stopped and the retryCounter param-
2353 eter reset. If all blocks in the current transmission window have been acknowledged, then the transmis-
2354 sion window shall be increased by *apscMaxWindowSize*. If all blocks have now been transmitted and acknowl-
2355 edged, then the transaction is complete, and an APSDE-DATA.confirm primitive shall be returned to the
2356 NHLE with a status value of SUCCESS. Otherwise, the block with the lowest unacknowledged block
2357 number shall be passed to the NWK data service.

2358 **2.2.8.4.5.2 Reception and Rejection, and Acknowledgements**

2359 If the fields required for a fragmentation-enabled transmission are not present in the frame it shall be re-
2360 jected. Also, any frames with parameters that fall outside the bounds of this protocol shall be rejected.

2361 If an incoming fragmented transaction is already in progress but the addressing and APS counter fields do
2362 not match those of the received frame, then the received frame may optionally be rejected or handled inde-
2363 pendently as a further transaction.

2364 If no transaction is in progress and a fragmented frame is received, then reassembly shall be attempted. Ini-
2365 tially the receive window shall be from 0 to (*apscMaxWindowSize*-1).

2366 If a transaction is initiated with APS counter and source address field values matching a previously re-
2367 ceived transaction, then the new transaction may be rejected as a duplicate.

2368 Upon receipt of the first received block (not necessarily block 0) in the first window, or when an acknowl-
2369 edgement is generated, the receiver shall set a timer for *apscAckWaitDuration*.

2370 If the receive window does not move forward within any (*apscAckWaitDuration* + *apscAckWaitDuration* *
2371 *apscMaxFrameRetries*) time period, the transaction shall be deemed to have failed. The receiver may send
2372 an acknowledgement to the sender with the block or blocks missed.

2373 If all blocks in the current receive window have been received and a block is received with a block number
2374 higher than the current receive window, then the receive window shall be increased by *apsMaxWindowSize*
2375 blocks.

2376 Additionally an APS acknowledgement shall be generated for the window if any one of the following cir-
2377 cumstances occurs: (1) the last block in the entire fragmented transmission is received, (2) the last block in
2378 the window is received, (3) a block is received and all subsequent blocks in the window have been previ-
2379 ously received and acknowledged. If a block is received with its block number value outside of the current
2380 window, then an acknowledgement shall NOT be generated.

2381 Once all blocks in the transaction have been received, the APS shall issue an APSDE-DATA.indication
 2382 primitive containing the reassembled message, and the transaction shall be deemed to be complete. A per-
 2383 iod of persistence of *apscAckWaitDuration* seconds is encouraged in order to facilitate any retranmission
 2384 of the final acknowledgement.

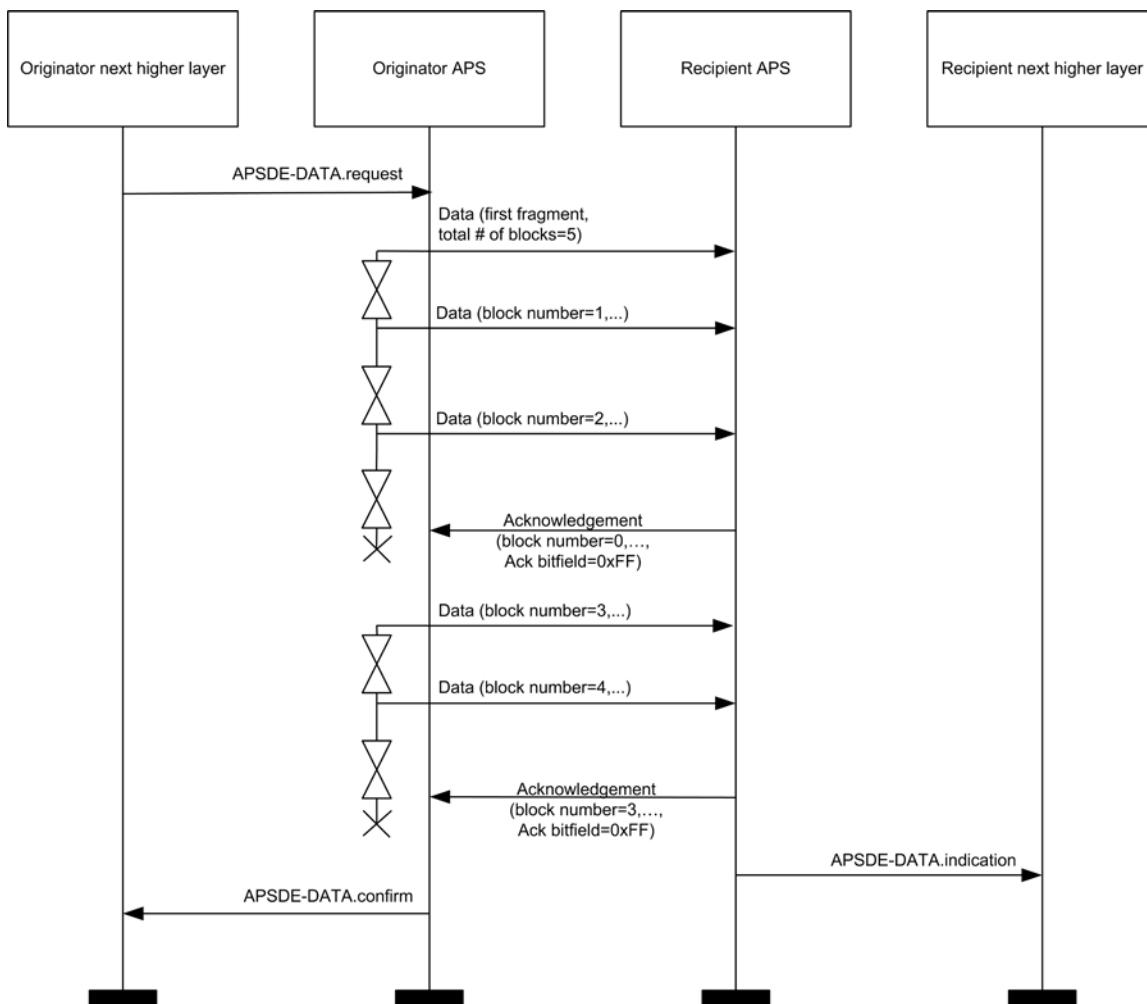
2385 Where generated, the acknowledgement is formatted according to the acknowledgement frame format
 2386 specified in section 2.2.5.2.3. The APS counter field shall reflect the value in the corresponding field of the
 2387 frame(s) being acknowledged. The block number field shall contain the value of the lowest block number
 2388 in the current receive window, using the value 0 as the value of the first block.

2389 The first bit of the ack bitfield shall be set to 1 if the first fragment in the current receive window has been
 2390 correctly received, and 0 otherwise. Subsequent bits shall be set similarly, with values corresponding to
 2391 subsequent fragments in the current receive window. If *apsMaxWindowSize* is less than 8, then the remain-
 2392 ing bits shall be set to 1.

2393 The process is illustrated in the following diagrams. In Figure 2.12, the transmission is successful immedi-
 2394 ately. (These examples assume that *apsMaxWindowSize* takes the value 3).

2395

Figure 2.12 Successful Data Transmission with Fragmentation

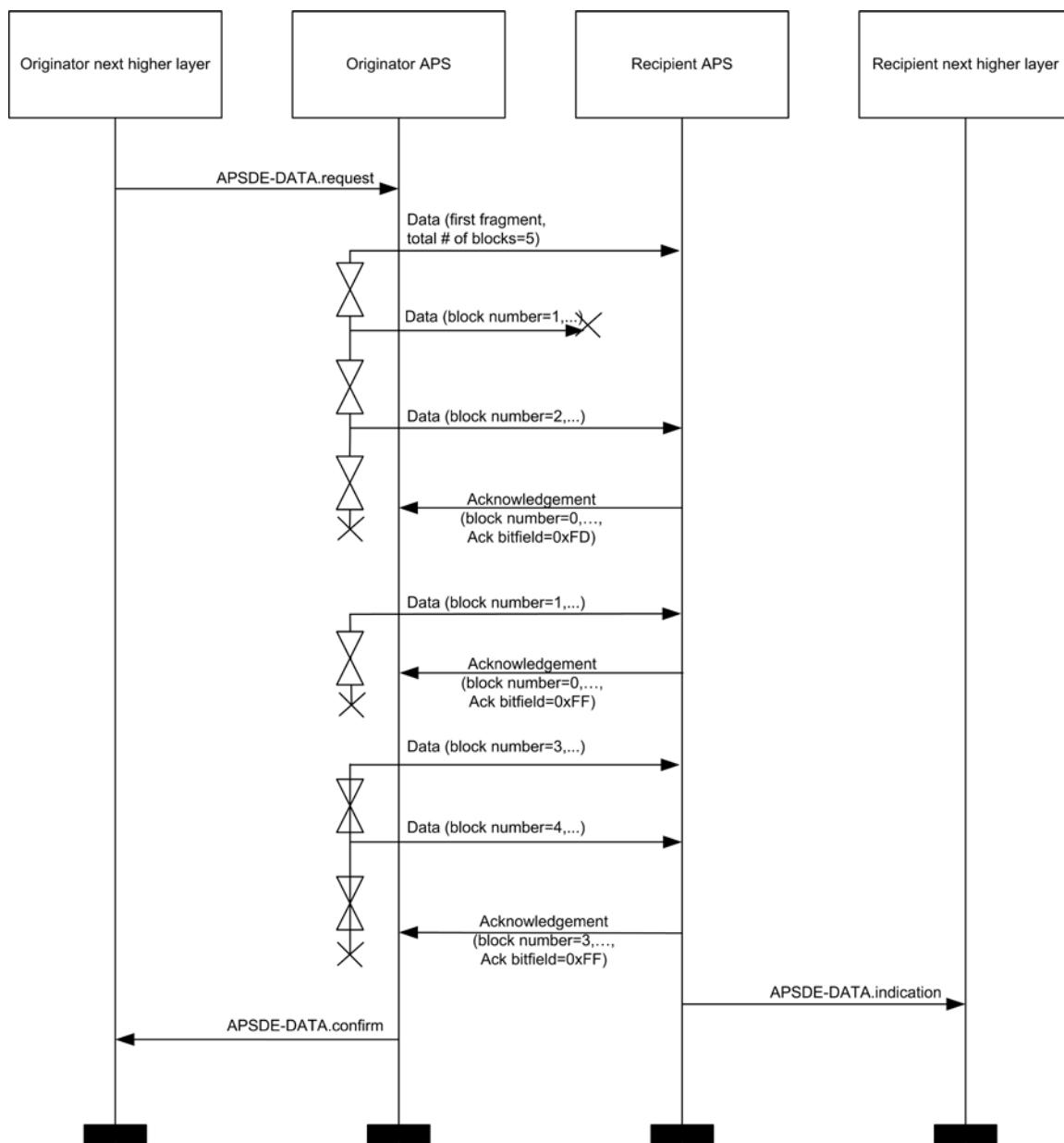


2396
 2397

2398 In Figure 2.13, a single frame is lost during transit across the network, and is retransmitted.

2399

Figure 2.13 Fragmented Data Transmission with a Single Retransmission



2400

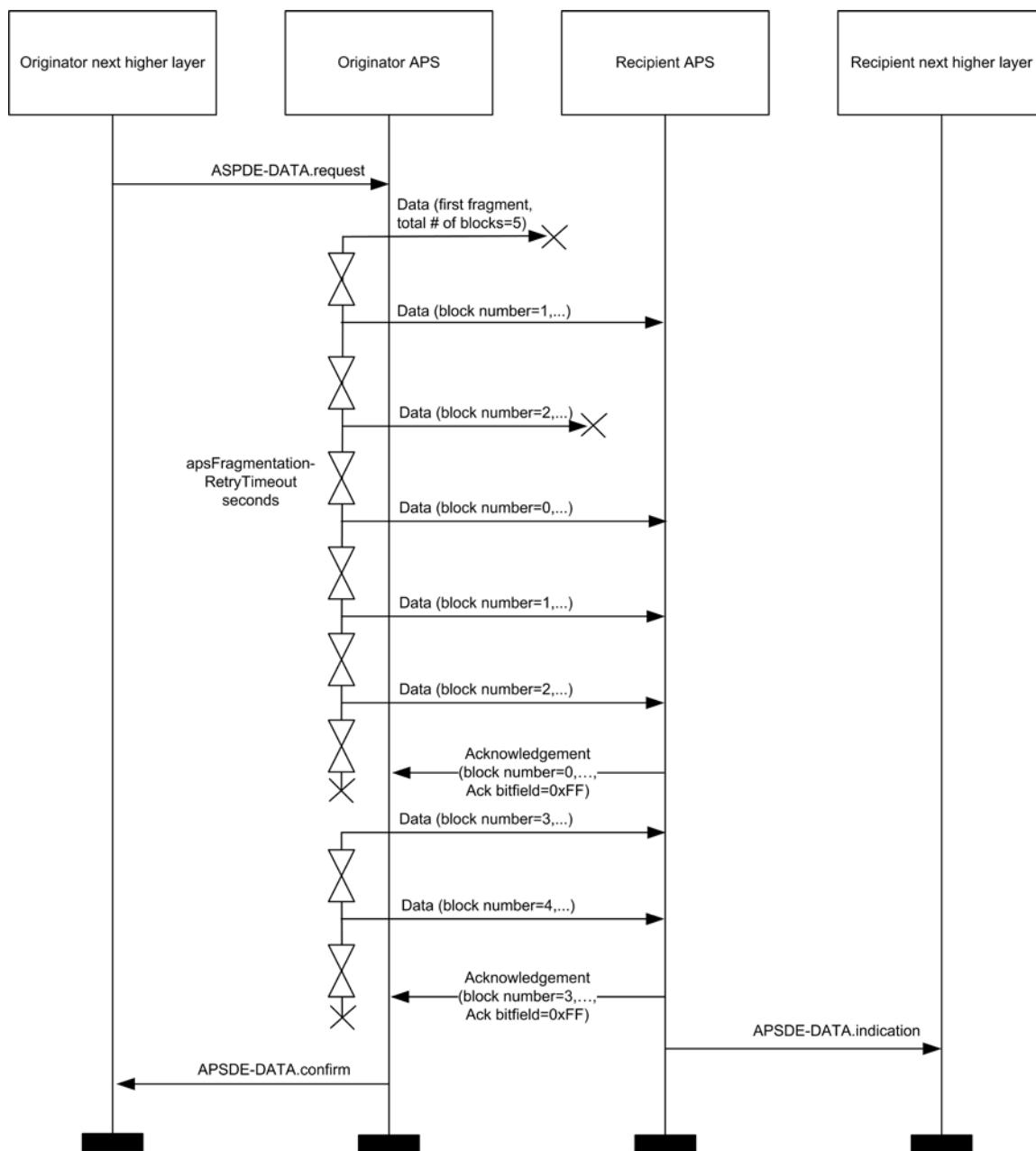
2401

2402
2403
2404

In Figure 2.14, multiple frames are lost in the network, including a frame which has the highest block number in the window. Slightly more traffic is required in this case, but the source backs off and gives the network a chance to recover, and the ASDU is delivered successfully.

2405

Figure 2.14 Fragmented Data Transmission with Multiple Retransmissions



2406

2407

2.2.9 APS Sub-Layer Status Values

2408
2409
2410

Application support (APS) sub-layer confirm primitives often include a parameter that reports on the status of the request to which the confirmation applies. Values for APS sub-layer Status parameters appear in Table 2.27.

Table 2.27 APS Sub-layer Status Values

Name	Value	Description
SUCCESS	0x00	A request has been executed successfully.
ASDU_TOO_LONG	0xa0	A transmit request failed since the ASDU is too large and fragmentation is not supported.
DEFRAG_DEFERRED	0xa1	A received fragmented frame could not be defragmented at the current time.
DEFRAG_UNSUPPORTED	0xa2	A received fragmented frame could not be defragmented since the device does not support fragmentation.
ILLEGAL_REQUEST	0xa3	A parameter value was out of range.
INVALID_BINDING	0xa4	An APSME-UNBIND.request failed due to the requested binding link not existing in the binding table.
INVALID_GROUP	0xa5	An APSME-REMOVE-GROUP.request has been issued with a group identifier that does not appear in the group table.
INVALID_PARAMETER	0xa6	A parameter value was invalid or out of range.
NO_ACK	0xa7	An APSDE-DATA.request requesting acknowledged transmission failed due to no acknowledgement being received.
NO_BOUND_DEVICE	0xa8	An APSDE-DATA.request with a destination addressing mode set to 0x00 failed due to there being no devices bound to this device.
NO_SHORT_ADDRESS	0xa9	An APSDE-DATA.request with a destination addressing mode set to 0x03 failed due to no corresponding short address found in the address map table.
NOT_SUPPORTED	0xaa	An APSDE-DATA.request with a destination addressing mode set to 0x00 failed due to a binding table not being supported on the device.
SECURED_LINK_KEY	0xab	An ASDU was received that was secured using a link key.
SECURED_NWK_KEY	0xac	An ASDU was received that was secured using a network key.

Name	Value	Description
SECURITY_FAIL	0xad	An APSDE-DATA.request requesting security has resulted in an error during the corresponding security processing.
TABLE_FULL	0xae	An APSME-BIND.request or APSME.ADD-GROUP.request issued when the binding or group tables, respectively, were full.
UNSECURED	0xaf	An ASDU was received without any security.
UNSUPPORTED_ATTRIBUTE	0xb0	An APSME-GET.request or APSME-SET.request has been issued with an unknown attribute identifier.

2412 **2.3 The ZigBee Application Framework**

2413 **2.3.1 Creating a ZigBee Profile**

2414 The key to communicating between devices on a ZigBee network is agreement on a profile.

2415 An example of a profile would be home automation. This ZigBee profile permits a series of device types to
2416 exchange control messages to form a wireless home automation application. These devices are designed to
2417 exchange well-known messages to effect control such as turning a lamp on or off, sending a light sensor
2418 measurement to a lighting controller, or sending an alert message if an occupancy sensor detects move-
2419 ment.

2420 An example of another type of profile is the device profile that defines common actions between ZigBee
2421 devices. To illustrate, wireless networks rely on the ability for autonomous devices to join a network and
2422 discover other devices and services on devices within the network. Device and service discovery are fea-
2423 tures supported within the device profile.

2424 **2.3.1.1 Getting a Profile Identifier from the ZigBee Alliance**

2425 ZigBee defines profiles in two separate classes: manufacturer-specific and public. The exact definition and
2426 criteria for these classes are an administrative issue within the ZigBee Alliance and outside the scope of this
2427 document. For the purposes of this technical specification, the only criterion is for profile identifiers to be
2428 unique. To that end, every profile effort must start with a request to the ZigBee Alliance for allocation of a
2429 profile identifier. Once the profile identifier is obtained, that profile identifier permits the profile designer
2430 to define the following:

- 2431 • Device descriptions
2432 • Cluster identifiers

2433 The application of profile identifiers to market space is a key criterion for issuance of a profile identifier
2434 from the ZigBee Alliance. The profile needs to cover a broad enough range of devices to permit interoper-
2435 ability to occur between devices, without being overly broad and resulting in a shortage of cluster identifiers
2436 to describe their interfaces. Conversely, the profile cannot be defined to be too narrowly, resulting in many
2437 devices described by individual profile identifiers, resulting in a waste of the profile identifier addressing
2438 space and interoperability issues in describing how the devices are interfaced. Policy groups within the
2439 ZigBee Alliance will establish criteria on how profiles are to be defined and to help requestors tailor their
2440 profile identifier requests.

2441 2.3.1.2 Defining Device Descriptions and Clusters

2442 The profile identifier is the main enumeration feature within the ZigBee protocol. Each unique profile iden-
2443 tifier defines an associated enumeration of device descriptions and cluster identifiers. For example, for pro-
2444 file identifier “1”, there exists a pool of device descriptions described by a 16-bit value (meaning there are
2445 65,536 possible device descriptions within each profile) and a pool of cluster identifiers described by a
2446 16-bit value (meaning there are 65,536 possible cluster identifiers within each profile). Each cluster identi-
2447 fier also supports a pool of attributes described by a 16-bit value. As such, each profile identifier has up to
2448 65,536 cluster identifiers and each of those cluster identifiers contains up to 65,536 attributes. It is the re-
2449 sponsibility of the profile developer to define and allocate device descriptions, cluster identifiers, and at-
2450 tributes within their allocated profile identifier. Note that the definition of device descriptions, cluster iden-
2451 tifiers, and attribute identifiers must be undertaken with care to ensure efficient creation of simple de-
2452 scriptors and simplified processing when exchanging messages.

2453 For public profile identifiers defined within the ZigBee Alliance, a cluster library has been created which
2454 provides a common definition and enumeration of clusters and their attributes. The cluster library is de-
2455 signed to sponsor re-use of cluster and attribute definitions across application profiles. By convention,
2456 when public profiles employ the cluster library, they will share a common enumeration and definition of
2457 cluster and attribute identifiers.

2458 Device descriptions and cluster identifiers must be accompanied by knowledge of the profile identifier to
2459 be processed. Prior to any messages being directed to a device, it is assumed by the ZigBee protocol that
2460 service discovery has been employed to determine profile support on devices and endpoints. Likewise, the
2461 binding process assumes similar service discovery and profile matching has occurred, since the resulting
2462 match is distilled to source address, source endpoint, cluster identifier, destination address, and destination
2463 endpoint.

2464 2.3.1.3 Deploying the Profile on Endpoints

2465 A single ZigBee device may contain support for many profiles, provide for subsets of various cluster iden-
2466 tifiers defined within those profiles, and may support multiple device descriptions. This capability is de-
2467 fined using a hierarchy of addressing within the device as follows:

- 2468 • **Device:** The entire device is supported by a single radio with a unique IEEE and NWK address.
- 2469 • **Endpoints:** This is an 8-bit field that describes different applications that are supported by a single ra-
2470 dio. Endpoint 0x00 is used to address the device profile, which each ZigBee device must employ,
2471 endpoint 0xff is used to address all active endpoints (the broadcast endpoint). Consequently, a single
2472 physical ZigBee radio can support up to 254 applications on endpoints 0x01-0xfe. Note that endpoints
2473 0xf1-0xfe can only be used for ZigBee Alliance approved applications.

2474 It is an application decision as to how to deploy applications on a device endpoint and which endpoints to
2475 advertise. The only requirement is that simple descriptors be created for each endpoint and those de-
2476 scriptors made available for service discovery.

2477 2.3.1.4 Enabling Service Discovery

2478 Once a device is created to support specific profiles and made consistent with cluster identifier usage for
2479 device descriptions within those profiles, the applications can be deployed. To do this, each application is
2480 assigned to individual endpoints and each described using simple descriptors (an endpoint can support only
2481 a single application profile). It is via the simple descriptors and other service discovery mechanisms de-
2482 scribed in the ZigBee device profile that service discovery is enabled, binding of devices is supported, and
2483 application messaging between complementary devices is facilitated.

2484 One important point is that service discovery is made on the basis of profile identifier, input cluster identi-
2485 fier list, and output cluster identifier list (device description is notably missing). The device description is
2486 simply a convention for specifying mandatory and optional cluster identifier support within devices of that
2487 type for the indicated profile. Additionally, it is expected that the device description enumeration would be
2488 employed within PDAs or other assisted binding devices to provide external descriptions of device capabili-
2489 ties.

2490 2.3.1.5 Mixing Standard and Proprietary Profiles

2491 As an example, a ZigBee device could be developed to ZigBee public profile identifier “XX.” If a manu-
2492 facturer wanted to deploy a ZigBee device supporting public profile “XX” and also provide manufacturer
2493 specific extensions, these extensions could be added to the manufacturer’s implementation of public profile
2494 “XX” if manufacturer extensions are supported within the definition of profile “XX.” Alternatively, if
2495 manufacturer extensions are not supported or the type of desired manufacturer extensions aren’t supported
2496 in profile “XX,” the manufacturer may deploy the extensions in a separate manufacturer-specific profile
2497 identifier advertised on a separate endpoint within the same physical device. In either case, devices that
2498 support the profile identifier “XX” but not containing the manufacturer extensions, would only advertise
2499 support for the base features of public profile identifier “XX” and could not respond to or create messages
2500 using the manufacturer extensions.

2501 2.3.1.6 Enabling Backward Compatibility

2502 In the previous example, a device is created using ZigBee public profile identifier “XX.” If the ZigBee Al-
2503 lliance were to update this public profile at a later time to add new features, the revisions could either be in-
2504 corporated directly into public profile identifier “XX” if such extensions are supported via the definition of
2505 the profile, or could be introduced into a new public profile with a new profile identifier (say “XY”). As-
2506 suming extensibility is not supported in public profile “XX,” devices manufactured with just profile identi-
2507 fier “XX” could still be compatible with newer devices manufactured later by having the newer devices
2508 advertise support for both profile identifier “XX” and profile identifier “XY.” In this manner, the newer
2509 device may communicate with older devices using profile identifier “XX”; however, it may also communi-
2510 cate with newer devices using profile identifier “XY” from within the same application. The service dis-
2511 covery feature within ZigBee enables devices on the network to determine the level of support.

2512 It is the goal of the ZigBee Alliance to provide extensibility, both for manufacturer extensions to public
2513 profiles as well as future enhancements to public profiles. That goal includes maintaining those extensions
2514 and enhancements within the same profile identifier whenever possible. This section illustrates that the pro-
2515 file definition features within ZigBee permit deployment of manufacturer extensions and feature enhance-
2516 ments, whether the goal of profile extensibility is achievable or not. The subject of profile extensibility,
2517 both for manufacturer extensions and feature enhancements, is beyond the scope of this document and ad-
2518 dressed in other Alliance documents.

2519 2.3.2 ZigBee Descriptors

2520 ZigBee devices describe themselves using descriptor data structures. The actual data contained in these de-
2521 scriptors is defined in the individual device descriptions. There are five descriptors: node, node power,
2522 simple, complex, and user, shown in Table 2.28.

2523 **Table 2.28 ZigBee Descriptors**

Descriptor Name	Status	Description
Node	M	Type and capabilities of the node.
Node power	M	Node power characteristics.
Simple	M	Device descriptions contained in node.
Complex	O	Further information about the device descriptions.
User	O	User-definable descriptor.

2524 **2.3.2.1 Transmission of Descriptors**

2525 The node, node power, simple, and user descriptors shall be transmitted in the order that they appear in
2526 their respective tables, i.e., the field at the top of the table is transmitted first and the field at the bottom of
2527 the table is transmitted last. Each individual field shall follow the transmission order specified in sec-
2528 tion 1.2.1.4.

2529 Each descriptor shall be less than or equal to *apscMaxDescriptorSize* unless provision has been made to
2530 enable transmission of discovery information without the mandatory use of fragmentation.

2531 In the case of the Simple Descriptor (see 2.3.2.5), transmission primitives exist which permit the descriptor
2532 to extend beyond *apscMaxDescriptorSize* (see 2.4.3.1.22 and 2.4.4.2.20). When extended transmission
2533 primitives are employed, the standard transmission primitives (see 2.4.3.1.5 and 2.4.4.2.5) require trans-
2534 mission of an abbreviated Simple Descriptor, and the Node Descriptor of the device shall indicate availa-
2535 bility of extended transmission primitives (see 2.3.2.3.12).

2536 The complex descriptor shall be formatted and transmitted as illustrated in Figure 2.15.

2537 **Figure 2.15 Format of the Complex Descriptor**

Octets: 1	Variable	...	Variable
Field count	Field 1	...	Field <i>n</i>

2538 Each field included in the complex descriptor shall be formatted as illustrated in Figure 2.16.

2539 **Figure 2.16 Format of an Individual Complex Descriptor Field**

Octets: 1	Variable
Compressed XML tag	Field data

2541 **2.3.2.1.1 Field Count Field**

2542 The field count field is one octet in length and specifies the number of fields included in the Complex De-
2543 scriptor, each formatted as illustrated in Figure 2.16.

2544 **2.3.2.1.1.1 Compressed XML Tag Field**

2545 The compressed XML tag field is one octet in length and specifies the XML tag for the current field. The
2546 compressed XML tags for the complex descriptor are listed in Table 2.41.

2547 **2.3.2.1.1.2 Field Data Field**

2548 The field data field has a variable length and contains the information specific to the current field, as indi-
2549 cated by the compressed XML tag field.

2550 **2.3.2.2 Discovery via Descriptors**

2551 Descriptor information is queried in the ZDO management entity device and service discovery, using the
2552 ZigBee device profile request primitive addressed to endpoint 0. For details of the discovery operation, see
2553 section 2.4.2.1. Information is returned via the ZigBee device profile indication primitive.

2554 The node, node power, complex, and user descriptors apply to the complete node. The simple descriptor
2555 must be specified for each endpoint defined in the node. If a node contains multiple subunits, these will be
2556 on separate endpoints and the specific descriptors for these endpoints are read by including the relevant
2557 endpoint number in the ZigBee device profile primitive.

2558 **2.3.2.3 Node Descriptor**

2559 The node descriptor contains information about the capabilities of the ZigBee node and is mandatory for
2560 each node. There shall be only one node descriptor in a node.

2561 The fields of the node descriptor are shown in Table 2.29 in their order of transmission.

2562 **Table 2.29 Fields of the Node Descriptor**

Field Name	Length (Bits)
Logical type	3
Complex descriptor available	1
User descriptor available	1
Reserved	3
APS flags	3
Frequency band	5
MAC capability flags	8
Manufacturer code	16
Maximum buffer size	8
Maximum incoming transfer size	16
Server mask	16
Maximum outgoing transfer size	16
Descriptor capability field	8

2563 **2.3.2.3.1 Logical Type Field**

2564 The logical type field of the node descriptor is three bits in length and specifies the device type of the
2565 ZigBee node. The logical type field shall be set to one of the non-reserved values listed in Table 2.30.

2566

Table 2.30 Values of the Logical Type Field

Logical Type Value $b_2b_1b_0$	Description
000	ZigBee coordinator
001	ZigBee router
010	ZigBee end device
011-111	Reserved

2567 **2.3.2.3.2 Complex Descriptor Available Field**

2568 The complex descriptor available field of the node descriptor is one bit in length and specifies whether a
 2569 complex descriptor is available on this device. If this field is set to 1, a complex descriptor is available. If
 2570 this field is set to 0, a complex descriptor is not available.

2571 **2.3.2.3.3 User Descriptor Available Field**

2572 The user descriptor available field of the node descriptor is one bit in length and specifies whether a user
 2573 descriptor is available on this device. If this field is set to 1, a user descriptor is available. If this field is set
 2574 to 0, a user descriptor is not available.

2575 **2.3.2.3.4 APS Flags Field**

2576 The APS flags field of the node descriptor is three bits in length and specifies the application support
 2577 sub-layer capabilities of the node.

2578 This field is currently not supported and shall be set to zero.

2579 **2.3.2.3.5 Frequency Band Field**

2580 The frequency band field of the node descriptor is five bits in length and specifies the frequency bands that
 2581 are supported by the underlying IEEE 802.15.4 radio utilized by the node. For each frequency band sup-
 2582 ported by the underlying IEEE 802.15.4 radio, the corresponding bit of the frequency band field, as listed in
 2583 Table 2.31, shall be set to 1. All other bits shall be set to 0.

2584 **Table 2.31 Values of the Frequency Band Field**

Frequency Band Field Bit Number	Supported Fre- quency Band
0	868 – 868.6 MHz
1	Reserved
2	902 – 928 MHz
3	2400 – 2483.5 MHz
4	Reserved

2585 **2.3.2.3.6 MAC Capability Flags Field**

2586 The MAC capability flags field is eight bits in length and specifies the node capabilities, as required by the
2587 IEEE 802.15.4-2003 MAC sub-layer [B1]. The MAC capability flags field shall be formatted as illustrated
2588 in Figure 2.17.

2589 **Figure 2.17 Format of the MAC Capability Flags Field**

Bits: 0	1	2	3	4-5	6	7
Alternate PAN coordinator	Device type	Power source	Receiver on when idle	Reserved	Security capability	Allocate address

2590
2591 The alternate PAN coordinator sub-field is one bit in length and shall be set to 1 if this node is capable of
2592 becoming a PAN coordinator. Otherwise, the alternative PAN coordinator sub-field shall be set to 0.

2593 The device type sub-field is one bit in length and shall be set to 1 if this node is a full function device
2594 (FFD). Otherwise, the device type sub-field shall be set to 0, indicating a reduced function device (RFD).

2595 The power source sub-field is one bit in length and shall be set to 1 if the current power source is mains
2596 power. Otherwise, the power source sub-field shall be set to 0. This information is derived from the node
2597 current power source field of the node power descriptor.

2598 The receiver on when idle sub-field is one bit in length and shall be set to 1 if the device does not disable its
2599 receiver to conserve power during idle periods. Otherwise, the receiver on when idle sub-field shall be set
2600 to 0 (see also section 2.3.2.4.)

2601 The security capability sub-field is one bit in length and shall be set to 1 if the device is capable of sending
2602 and receiving frames secured using the security suite specified in [B1]. Otherwise, the security capability
2603 sub-field shall be set to 0.

2604 The allocate address sub-field is one bit in length and shall be set to 0 or 1.

2605 **2.3.2.3.7 Manufacturer Code Field**

2606 The manufacturer code field of the node descriptor is sixteen bits in length and specifies a manufacturer
2607 code that is allocated by the ZigBee Alliance, relating the manufacturer to the device.

2608 **2.3.2.3.8 Maximum Buffer Size Field**

2609 The maximum buffer size field of the node descriptor is eight bits in length, with a valid range of
2610 0x00-0x7f. This field specifies the maximum size, in octets, of the network sub-layer data unit (NSDU) for
2611 this node. This is the maximum size of data or commands passed to or from the application by the applica-
2612 tion support sub-layer, before any fragmentation or re-assembly.

2613 This field can be used as a high-level indication for network management.

2614 **2.3.2.3.9 Maximum Incoming Transfer Size Field**

2615 The maximum transfer size field of the node descriptor is sixteen bits in length, with a valid range of
2616 0x0000-0x7fff. This field specifies the maximum size, in octets, of the application sub-layer data unit
2617 (ASDU) that can be transferred to this node in one single message transfer. This value can exceed the value
2618 of the node maximum buffer size field (see section 2.3.2.3.8) through the use of fragmentation.

2619 **2.3.2.3.10 Server Mask Field**

2620 The server mask field of the node descriptor is sixteen bits in length, with bit settings signifying the system
2621 server capabilities of this node. It is used to facilitate discovery of particular system servers by other nodes
2622 on the system. The bit settings are defined in Table 2.32.

2623

Table 2.32 Server Mask Bit Assignments

Bit Number	Assignment
0	Primary Trust Center
1	Backup Trust Center
2	Primary Binding Table Cache
3	Backup Binding Table Cache
4	Primary Discovery Cache
5	Backup Discovery Cache
6	Network Manager
7 – 8	Reserved
9 – 15	Stack Compliance Revision

2624

2.3.2.3.10.1 Stack Compliance Revision

2625 These bits indicate the revision of the ZigBee Pro Core specification that the running stack is implemented to.
2626 Prior to revision 21 of the specification these bits were reserved and thus set to 0. A stack that is compliant
2627 to revision 21 would set these bits to 21 (0010101b). A stack shall indicate the revision of the specification
2628 it is compliant to by setting these bits.

2630

2.3.2.3.11 Maximum Outgoing Transfer Size Field

2631 The maximum transfer size field of the node descriptor is sixteen bits in length, with a valid range of
2632 0x0000-0x7fff. This field specifies the maximum size, in octets, of the application sub-layer data unit
2633 (ASDU) that can be transferred from this node in one single message transfer. This value can exceed the
2634 value of the node maximum buffer size field (see section 2.3.2.3.8) through the use of fragmentation.

2.3.2.3.12 Descriptor Capability Field

2635 The descriptor capability field of the node descriptor is eight bits in length, with bit settings signifying the
2636 descriptor capabilities of this node. It is used to facilitate discovery of particular features of the descriptor
2637 fields by other nodes on the system. The bit settings are defined in Table 2.33.

2640

Table 2.33 Descriptor Capability Bit Assignments

Bit Number	Assignment
0	Extended Active Endpoint List Available
1	Extended Simple Descriptor List Available

Bit Number	Assignment
2–7	Reserved

2641 2.3.2.4 Node Power Descriptor

2642 The node power descriptor gives a dynamic indication of the power status of the node and is mandatory for
2643 each node. There shall be only one node power descriptor in a node.

2644 The fields of the node power descriptor are shown in Table 2.34 in the order of their transmission.

2645 **Table 2.34 Fields of the Node Power Descriptor**

Field Name	Length (Bits)
Current power mode	4
Available power sources	4
Current power source	4
Current power source level	4

2646 2.3.2.4.1 Current Power Mode Field

2647 The current power mode field of the node power descriptor is four bits in length and specifies the current
2648 sleep/power-saving mode of the node. The current power mode field shall be set to one of the non-reserved
2649 values listed in Table 2.35.

2650 **Table 2.35 Values of the Current Power Mode Field**

Current Power Mode Value $b_3b_2b_1b_0$	Description
0000	Receiver synchronized with the receiver on when idle subfield of the node descriptor.
0001	Receiver comes on periodically as defined by the node power descriptor.
0010	Receiver comes on when stimulated, for example, by a user pressing a button.
0011-1111	Reserved.

2651 2.3.2.4.2 Available Power Sources Field

2652 The available power sources field of the node power descriptor is four bits in length and specifies the power
2653 sources available on this node. For each power source supported on this node, the corresponding bit of
2654 the available power sources field, as listed in Table 2.36, shall be set to 1. All other bits shall be set to 0.

2655

Table 2.36 Values of the Available Power Sources Field

Available Power Sources Field Bit Number	Supported Power Source
0	Constant (mains) power
1	Rechargeable battery
2	Disposable battery
3	Reserved

2656 **2.3.2.4.3 Current Power Source Field**

2657 The current power source field of the node power descriptor is four bits in length and specifies the current
2658 power source being utilized by the node. For the current power source selected, the corresponding bit of the
2659 current power source field, as listed in Table 2.37, shall be set to 1. All other bits shall be set to 0.

2660

Table 2.37 Values of the Current Power Sources Field

Current Power Source Field Bit Number	Current Power Source
0	Constant (mains) power
1	Rechargeable battery
2	Disposable battery
3	Reserved

2661 **2.3.2.4.4 Current Power Source Level Field**

2662 The current power source level field of the node power descriptor is four bits in length and specifies the
2663 level of charge of the power source. The current power source level field shall be set to one of the
2664 non-reserved values listed in Table 2.38.

2665

Table 2.38 Values of the Current Power Source Level Field

Current Power Source Level Field $b_3b_2b_1b_0$	Charge Level
0000	Critical
0100	33%
1000	66%
1100	100%
All other values	Reserved

2666 2.3.2.5 Simple Descriptor

2667 The simple descriptor contains information specific to each endpoint contained in this node. The simple
 2668 descriptor is mandatory for each endpoint present in the node.

2669 The fields of the simple descriptor are shown in Table 2.39 in their order of transmission. As this descriptor
 2670 needs to be transmitted over air, the overall length of the simple descriptor shall be less than or equal to
 2671 $apscMaxDescriptorSize$.

2672

Table 2.39 Fields of the Simple Descriptor

Field Name	Length (Bits)
Endpoint	8
Application profile identifier	16
Application device identifier	16
Application device version	4
Reserved	4
Application input cluster count	8
Application input cluster list	$16*i$ (where i is the value of the application input cluster count)
Application output cluster count	8
Application output cluster list	$16*o$ (where o is the value of the application output cluster count)

2673 **2.3.2.5.1 Endpoint Field**

2674 The endpoint field of the simple descriptor is eight bits in length and specifies the endpoint within the node
2675 to which this description refers. Applications shall only use endpoints 1-254. Endpoints 241-254 shall be
2676 used only with the approval of the ZigBee Alliance. The Green Power cluster, if implemented, shall use
2677 endpoint 242.

2678 **2.3.2.5.2 Application Profile Identifier Field**

2679 The application profile identifier field of the simple descriptor is sixteen bits in length and specifies the profile
2680 that is supported on this endpoint. Profile identifiers shall be obtained from the ZigBee Alliance.

2681 **2.3.2.5.3 Application Device Identifier Field**

2682 The application device identifier field of the simple descriptor is sixteen bits in length and specifies the de-
2683 vice description supported on this endpoint. Device description identifiers shall be obtained from the
2684 ZigBee Alliance.

2685 **2.3.2.5.4 Application Device Version Field**

2686 The application device version field of the simple descriptor is four bits in length and specifies the version
2687 of the device description supported on this endpoint. The application device version field shall be set to one
2688 of the non-reserved values listed in Table 2.40.

2689 **Table 2.40 Values of the Application Device Version Field**

Application Device Version Value b ₃ b ₂ b ₁ b ₀	Description
0000-1111	Specific values to be set by the application profile described by the application profile identifier in this descriptor. Default shall be 0000 unless otherwise defined by the application profile.

2690 **2.3.2.5.5 Application Input Cluster Count Field**

2691 The application input cluster count field of the simple descriptor is eight bits in length and specifies the number
2692 of input clusters, supported on this endpoint that will appear in the application input cluster list field. If the value of this field is zero, the application input cluster list field shall not be included.
2693

2694 **2.3.2.5.6 Application Input Cluster List**

2695 The application input cluster list of the simple descriptor is $16*i$ bits in length, where i is the value of the application input cluster count field. This field specifies the list of input clusters supported on this endpoint, for use during the service discovery and binding procedures.
2696
2697

2698 The application input cluster list field shall be included only if the value of the application input cluster count field is greater than zero.
2699

2700 **2.3.2.5.7 Application Output Cluster Count Field**

2701 The application output cluster count field of the simple descriptor is eight bits in length and specifies the number
2702 of output clusters, supported on this endpoint that will appear in the application output cluster list field. If the value of this field is zero, the application output cluster list field shall not be included.
2703

2704 **2.3.2.5.8 Application Output Cluster List**

2705 The application output cluster list of the simple descriptor is $16*o$ bits in length, where o is the value of the application output cluster count field. This field specifies the list of output clusters supported on this endpoint, for use during the service discovery and binding procedures.
2706
2707

2708 The application output cluster list field shall be included only if the value of the application output cluster
2709 count field is greater than zero.

2710 **2.3.2.6 Complex Descriptor**

2711 The complex descriptor contains extended information for each of the device descriptions contained in this
2712 node. The use of the complex descriptor is optional.

2713 Due to the extended and complex nature of the data in this descriptor, it is presented in XML form using
2714 compressed XML tags. Each field of the descriptor, shown in Table 2.41, can therefore be transmitted in
2715 any order. As this descriptor needs to be transmitted over air, the overall length of the complex descriptor
2716 shall be less than or equal to *apscMaxDescriptorSize*.

2717 **Table 2.41 Fields of the Complex Descriptor**

Field Name	XML Tag	Compressed XML Tag Value x_1x_0	Data Type
Reserved	-	00	-
Language and character set	<languageChar>	01	See section 2.3.2.6.1
Manufacturer name	<manufacturerName>	02	Character string
Model name	<modelName>	03	Character string
Serial number	<serialNumber>	04	Character string
Device URL	<deviceURL>	05	Character string
Icon	<icon>	06	Octet string
Icon URL	<outliner>	07	Character string
Reserved	-	08 – ff	-

2718 **2.3.2.6.1 Language and Character Set Field**

2719 The language and character set field is three octets in length and specifies the language and character set
2720 used by the character strings in the complex descriptor. The format of the language and character set field is
2721 illustrated in Figure 2.18.

2722 **Figure 2.18 Format of the Language and Character Set Field**

Octets: 2	1
ISO 639-1 language code	Character set identifier

2723

2724 The ISO 639-1 language code sub-field is two octets in length and specifies the language used for character
2725 strings, as defined in [B5].

2726 The character set identifier sub-field is one octet in length and specifies the encoding used by the characters
2727 in the character set. This sub-field shall be set to one of the non-reserved values listed in Table 2.42.

2728 **Table 2.42 Values of the Character Set Identifier Sub-Field**

Character Set Identifier Value	Bits Per Character	Description
0x00	8	ISO 646, ASCII character set. Each character is fitted into the least significant 7 bits of an octet with the most significant bit set to zero (see also [B6]).
0x01 – 0xff	-	Reserved.

2729
2730 If the language and character sets have not been specified, the language shall default to English (language
2731 code = “EN”) and the character set to ISO 646.

2.3.2.6.2 **Manufacturer Name Field**

2733 The manufacturer name field has a variable length and contains a character string representing the name of
2734 the manufacturer of the device.

2.3.2.6.3 **Model Name Field**

2736 The model name field has a variable length and contains a character string representing the name of the
2737 manufacturer’s model of the device.

2.3.2.6.4 **Serial Number Field**

2739 The serial number field has a variable length and contains a character string representing the manufacturer’s
2740 serial number of the device.

2.3.2.6.5 **Device URL Field**

2742 The device URL field has a variable length and contains a character string representing the URL through
2743 which more information relating to the device can be obtained.

2.3.2.6.6 **Icon Field**

2745 The icon field has a variable length and contains an octet string which carries the data for an icon that can
2746 represent the device on a computer, gateway, or PDA. The format of the icon shall be a 32-by-32-pixel
2747 PNG image.

2.3.2.6.7 **Icon URL Field**

2749 The icon URL field has a variable length and contains a character string representing the URL through
2750 which the icon for the device can be obtained.

2.3.2.7 **User Descriptor**

2752 The user descriptor contains information that allows the user to identify the device using a user-friendly
2753 character string, such as “Bedroom TV” or “Stairs light”. The use of the user descriptor is optional. This
2754 descriptor contains a single field, which uses the ASCII character set, and shall contain a maximum of 16
2755 characters.

2756 The fields of the user descriptor are shown in Table 2.43 in the order of their transmission.

2757

Table 2.43 Fields of the User Descriptor

Field Name	Length (Octets)
User description	16

2758

2.3.3 Functional Description

2759

2.3.3.1 Reception and Rejection

2760
2761

The application framework shall be able to filter frames arriving via the APS sub-layer data service and only present the frames that are of interest to the applications implemented on each active endpoint.

2762
2763
2764

The application framework receives data from the APS sub-layer via the APSDE-DATA.indication primitive and is targeted at a specific endpoint (DstEndpoint parameter) and a specific profile (ProfileId parameter).

2765
2766
2767
2768

If the application framework receives a frame for an inactive endpoint, the frame shall be discarded. Otherwise, if the profile identifier passes the Profile Id Endpoint Matching Rules (see section 2.3.3.2), the application framework shall pass the payload of the received frame to the application implemented on the specified endpoint.

2769

2.3.3.2 Profile ID Endpoint Matching Rules

2770
2771

Table 2.44 below details the matching of incoming APS datagrams or ZDO discovery messages are matched.

2772

2773

Table 2.44 Profile ID Endpoint Matching Rules

Incoming Message	APS or ZDO Profile ID	Destination Endpoint SimpleDescriptor							
		ZDO	Legacy	Common	ZSE	GW	MSP	GP	ZLL
ZDO	0x0000	ZDO	X	X	X	X	X	X	X
Legacy	0x0101 – 0x0103, 0x0105 – 0x0108	X	Legacy	Legacy	X	X	X	X	X
Common (HA)	0x0104	X	X	Common	X	X	X	X	X
ZSE	0x0109	X	X	X	ZSE	X	X	X	X
Gateway (GW)	0x7F02	X	X	X	X	GW	X	X	X
MSP	0x8000 – 0xFF00, 0x7F01	X	X	X	X	X	MSP	X	X
GreenPower (GP)	0xA1E0	X	X	X	X	X	X	GP	X
ZLL	0xC05E	X	X	ZLL	X	X	X	X	ZLL
Wildcard	0xFFFF	ZDO	Legacy	Common	ZSE	GW	X	X	ZLL

2774

2775

2.3.3.2.1 Profile ID Endpoint Matching Rules for Incoming Messages

2776

2777 To apply Profile ID Endpoint matching rules for an incoming message, perform the following:

2778 (1) Starting on the Left side of the table, find the row that matches the profile ID of the incoming
2779 message.

2780 (2) If no match is found, then the message shall be dropped and no further processing shall take place.

2781 (3) Lookup the Simple Descriptor of the local destination Endpoint.

2782 (4) If no Simple Descriptor exists for the local destination endpoint, the message shall be dropped and
2783 no further processing shall be done.

2784 (5) If a Simple Descriptor exists, follow across the selected row in the table to where the Profile ID at
2785 the top of the column matches the Profile ID of the simple descriptor of the destination endpoint.

2786 (6) If an ‘x’ appears in the selected row, then the message shall be dropped and no further processing
2787 shall take place.

2788 (7) If a value other than ‘x’ appears in the selected row, then the message shall be processed. The
2789 value in the cell indicates the Profile ID that shall be used for any response message generated.
2790 If a range of values exist (for example Legacy), then the exact value for the Profile ID on the in-
2791 coming message may be used on any outgoing message generated.

2792

2793 For ZDO messages, the Profile ID Endpoint matching may be applied twice. The first time the rules will
2794 be applied on the message as a normal incoming APS datagram. For certain ZDO messages, the rules will
2795 be applied again to determine if the contents of the ZDO message match.

2796

2.3.3.2.2 Profile ID Endpoint Matching Rules for ZDO Contents

2798 To apply Profile ID Endpoint matching rules on the contents of ZDO discovery messages, perform the fol-
2799 lowing:

- 2800 (1) Starting on the left side of the table, find the row that matches the profile ID within the payload of
2801 the ZDO message (do not consider the Profile ID of the incoming ZDO message, which is always
2802 0x0000).
- 2803 (2) If no match is found, then there is no match for the discovery. Do the following:
- 2804 (a) Return an empty list of endpoints to the ZDO for processing. A response may be generated
2805 according to the rules of ZDO discovery. No further match processing on the message
2806 shall take place.
- 2807 (3) If a match is found, lookup the Simple Descriptor for all local endpoints. For each simple de-
2808 scription, perform the following:
- 2809 (a) Follow the previously selected row across the table and find the column with a Profile ID that
2810 matches the Simple Descriptor.
- 2811 (b) If a column with a matching Profile ID does not exist, then there is no match. Continue
2812 processing on the next local endpoint.
- 2813 (c) If the Profile ID at the top of the column matches, examine the contents of the cell.
- 2814 (d) If an X is found in the cell, then there is no match. Continue processing on the next local
2815 endpoint.
- 2816 (e) If a value other than X is found in the table, then a match exists. Add the endpoint and the
2817 associated Profile ID of the simple descriptor to the list of matches.
- 2818 (4) Once all endpoints have been analyzed, return the list of matching endpoints and the associated
2819 Profile IDs for each endpoint to the ZDO for processing.

2820 2.4 The ZigBee Device Profile

2821 2.4.1 Scope

2822 This ZigBee Application Layer Specification describes how general ZigBee device features such as Binding,
2823 Device Discovery, and Service Discovery are implemented within ZigBee Device Objects. The ZigBee
2824 Device Profile operates like any ZigBee profile by defining clusters. Unlike application specific profiles,
2825 the clusters within the ZigBee Device Profile define capabilities supported in all ZigBee devices. As with
2826 any profile document, this document details the mandatory and/or optional clusters.

2827 2.4.2 Device Profile Overview

2828 The Device Profile supports four key inter-device communication functions within the ZigBee protocol.
2829 These functions are explained in the following sections:

- 2830 • Device and Service Discovery Overview
- 2831 • End Device Bind Overview
- 2832 • Bind and Unbind Overview
- 2833 • Binding Table Management Overview
- 2834 • Network Management Overview

2835 2.4.2.1 Device and Service Discovery Overview

2836 Device and Service Discovery are distributed operations where individual devices or designated discovery
2837 cache devices respond to discovery requests. The “device address of interest” field enables responses from
2838 either the device itself or a discovery cache device. In selected cases where both the discovery cache device
2839 or the device’s parent and the “device address of interest” device respond, the response from the “device
2840 address of interest” shall be used.

2841 The following capabilities exist for device and service discovery:

- 2842 • **Device Discovery:** Provides the ability for a device to determine the identity of other devices on the
2843 PAN. Device Discovery is supported for both the 64-bit IEEE address and the 16-bit Network address.
 - 2844 ○ Device Discovery messages can be used in one of two ways:
 - 2845 — **Broadcast addressed:** All devices on the network shall respond according to the Logical De-
2846 vice Type and the matching criteria. ZigBee End Devices shall respond with just their ad-
2847 dress. ZigBee Coordinators and ZigBee Routers with associated devices shall respond with
2848 their address as the first entry followed by the addresses of their associated devices depending
2849 on the type of request. The responding devices shall employ APS acknowledged service on
2850 the unicast responses.
 - 2851 — **Unicast addressed:** Only the specified device responds. A ZigBee End Device shall respond
2852 only with its address. A ZigBee Coordinator or Router shall reply with its own address and
2853 the address of each associated child device. Inclusion of the associated child devices allows
2854 the requestor to determine the network topology underlying the specified device.
 - 2855 • **Service Discovery:** Provides the ability for a device to determine services offered by other devices on
2856 the PAN.
 - 2857 ○ Service Discovery messages can be used in one of two ways:
 - 2858 — **Broadcast addressed:** Due to the volume of information that could be returned, only the in-
2859 dividual device or the primary discovery cache shall respond with the matching criteria es-
2860 tablished in the request. The primary discovery cache shall only respond in this case if it holds
2861 cached discovery information for the NWKAddrOfInterest from the request. The responding
2862 devices shall also employ APS acknowledged service on the unicast responses.

- 2863 — **Unicast addressed:** Only the specified device shall respond. In the case of a ZigBee Coordinator or ZigBee Router, these devices shall cache the Service Discovery information for sleeping associated devices and respond on their behalf.
- 2864 ○ Service Discovery is supported with the following query types:
- 2865 — **Active Endpoint:** This command permits an enquiring device to determine the active endpoints. An active endpoint is one with an application supporting a single profile, described by a Simple Descriptor. The command shall be unicast addressed.
- 2866 — **Match Simple Descriptor:** This command permits enquiring devices to supply a Profile ID (and, optionally, lists of input and/or output Cluster IDs) and ask for a return of the identity of an endpoint on the destination device which matches the supplied criteria. This command may be broadcast to all devices for which macRxOnWhenIdle = TRUE, or unicast addressed. For broadcast addressed requests, the responding device shall employ APS acknowledged service on the unicast responses.
- 2867 — **Simple Descriptor:** This command permits an enquiring device to return the Simple Descriptor for the supplied endpoint. This command shall be unicast addressed.
- 2868 — **Node Descriptor:** This command permits an enquiring device to return the Node Descriptor from the specified device. This command shall be unicast addressed.
- 2869 — **Power Descriptor:** This command permits an enquiring device to return the Power Descriptor from the specified device. This command shall be unicast addressed.
- 2870 — **Complex Descriptor:** This optional command permits an enquiring device to return the Complex Descriptor from the specified device. This command shall be unicast addressed.
- 2871 — **User Descriptor:** This optional command permits an enquiring device to return the User Descriptor from the specified device. This command shall be unicast addressed.

2886 **2.4.2.2 End Device Bind Overview**

2887 The following capabilities exist for end device bind:

- 2888 • **End Device Bind:**
 - 2889 ○ Provides the ability for an application to support a simplified method of binding where user intervention is employed to identify command/control device pairs. Typical usage would be where a user is asked to push buttons on two devices for installation purposes. Using this mechanism a second time allows the user to remove the binding table entry.

2893 **2.4.2.3 Bind and Unbind Overview**

2894 The following capabilities exist for directly configuring binding table entries:

- 2895 • **Bind:** provides the ability for creation of a Binding Table entry that maps control messages to their intended destination.
- 2896 • **Unbind:** provides the ability to remove Binding Table entries.

2898 **2.4.2.4 Binding Table Management Overview**

2899 The following capabilities exist for management of binding tables:

- 2900 • Registering devices that implement source binding:
 - 2901 ○ Provides the ability for a source device to instruct its primary binding table cache to hold its own binding table.
- 2903 • Replacing a device with another wherever it occurs in the binding table:
 - 2904 ○ Provides the ability to replace one device for another, by replacing all instances of its address in the binding table.
- 2906 • Backing up a binding table entry:

- 2907 ○ Provides the ability for a primary binding table cache to send details of a newly created entry to the
2908 backup binding table cache (after receiving a bind request).
- 2909 ● Removing a backup binding table entry:
 - 2910 ○ Provides the ability for a primary binding table cache to request that a specific entry be removed
2911 from the backup binding table cache (after receiving an unbind request).
- 2912 ● Backing up of the entire binding table:
 - 2913 ○ Provides the ability for a primary binding table cache to request backup of its entire binding table,
2914 using the backup binding table cache.
- 2915 ● Restoring the entire binding table:
 - 2916 ○ Provides the ability for a primary binding table cache to request restoration of its entire binding
2917 table, using the backup binding table cache.
- 2918 ● Backing up the Primary Binding Table Cache:
 - 2919 ○ Provides the ability for a primary binding table cache to request backup of its entire source devices
2920 address table (which contains the addresses of any source device containing its own binding table).
- 2921 ● Restoring the Primary Binding Table Cache:
 - 2922 ○ Provides the ability for a primary binding table cache to request restoration of its entire source de-
2923 vices address table (which contains the addresses of any source device containing its own binding
2924 table).

2925 2.4.2.5 Network Management Overview

2926 The following capabilities exist for network management:

- 2927 • Provides the ability to retrieve management information from the devices including:
 - 2928 o Network discovery results
 - 2929 o Link quality to neighbor nodes
 - 2930 o Routing table contents
 - 2931 o Binding table contents
 - 2932 o Discovery cache contents
 - 2933 o Energy detection scan results
- 2934 • Provides the ability to set management information controls including:
 - 2935 o Network leave
 - 2936 o Network direct join
 - 2937 o Permit joining
 - 2938 o Network update and fault notification

2939 2.4.2.6 Device Descriptions for the Device Profile

2940 The ZigBee Device Profile utilizes a single Device Description. Each cluster specified as Mandatory shall
2941 be present in all ZigBee devices. The response behavior to some messages is logical device type specific.
2942 The support for optional clusters is not dependent on the logical device type.

2943 2.4.2.7 Configuration and Roles

2944 The Device Profile assumes a client/server topology. A device making Device Discovery, Service Discovery,
2945 Binding or Network Management requests does so via a client role. A device which services these re-
2946 quests and responds does so via a server role. The client and server roles are non-exclusive in that a given
2947 device may supply both client and server roles.

2948 Since many client requests and server responses are public and accessible to application objects other than
2949 ZigBee Device Objects, the Transaction Sequence number in the Application Framework header shall be
2950 the same on client requests and their associated server responses.

2951 The Device Profile describes devices in one of two configurations:

- 2952 • **Client:** A client issues requests to the server via Device Profile messages.
- 2953 • **Server:** A server issues responses to the client that initiated the Device Profile message.

2954 2.4.2.8 Transmission of ZDP Commands

2955 All ZDP commands shall be transmitted via the APS data service and shall be formatted according to the
2956 ZDP frame structure, as illustrated in Figure 2.19.

2957 **Figure 2.19 Format of the ZDP Frame**

Octets: 1	Variable
Transaction sequence number	Transaction data

2958 **2.4.2.8.1 Transaction Sequence Number Field**

2959 The transaction sequence number field is eight bits in length and specifies an identification number for the
2960 ZDP transaction so that a response command frame can be related to the request frame. The application
2961 object itself shall maintain an eight-bit counter that is copied into this field and incremented by one for each
2962 command sent. When a value of 0xff is reached, the next command shall restart the counter with a value of
2963 0x00.

2964 If a device sends a ZDP request command that requires a response, the target device shall respond with the
2965 relevant ZDP response command and include the transaction sequence number contained in the original
2966 request command.

2967 The transaction sequence number field can be used by a controlling device, which may have issued multi-
2968 ple commands, so that it can match the incoming responses to the relevant command.

2969 **2.4.2.8.2 Transaction Data Field**

2970 The transaction data field has a variable length and contains the data for the individual ZDP transaction.
2971 The format and length of this field is dependent on the command being transmitted, as defined in sections
2972 2.4.3 and 2.4.4.

2973 **2.4.3 Client Services**

2974 The Device Profile Client Services support the transport of device and service discovery requests, end de-
2975 vice binding requests, bind requests, unbind requests, and network management requests from client to
2976 server. Additionally, Client Services support receipt of responses to these requests from the server.

2977 **2.4.3.1 Device and Service Discovery Client Services**

2978 Table 2.45 lists the commands supported by Device Profile, Device, and Service Discovery Client Services.
2979 Each of these commands will be discussed in the following sections.

2980 **Table 2.45 Device and Service Discovery Client Services Commands**

Device and Service Discovery Client Services	Client Transmission	Server Processing
NWK_addr_req	O	M
IEEE_addr_req	O	M
Node_Desc_req	M	M
Power_Desc_req	O	M
Simple_Desc_req	O	M
Active_EP_req	O	M
Match_Desc_req	O	M
Complex_Desc_req	O	O
User_Desc_req	O	O

Device and Service Discovery Client Services	Client Transmission	Server Processing
Discovery_Cache_req	O	O
Device_annce	O	M
Parent_annce	M	M
Parent_annce_rsp	M	M
User_Desc_set	O	O
System_Server_Discover_req	O	O
Discovery_store_req	O	O
Node_Desc_store_req	O	O
Power_Desc_store_req	O	O
Active_EP_store_req	O	O
Simple_Desc_store_req	O	O
Remove_node_cache_req	O	O
Find_node_cache_req	O	O
Extended_Simple_Desc_req	O	O
Extended_Active_EP_req	O	O

2981 **2.4.3.1.1 NWK_addr_req**

2982 The NWK_addr_req command (ClusterID=0x0000) shall be formatted as illustrated in Figure 2.20.

2983 **Figure 2.20 Format of the NWK_addr_req Command**

Octets: 8	1	1
IEEEAddress	RequestType	StartIndex

2984

2985 Table 2.46 specifies the fields of the NWK_addr_req Command Frame.

2986 **Table 2.46 Fields of the NWK_addr_req Command**

Name	Type	Valid Range	Description
IEEEAddr	IEEE Address	A valid 64-bit IEEE address	The IEEE address to be matched by the Remote Device
RequestType	Integer	0x00-0xff	Request type for this command: 0x00 – Single device response 0x01 – Extended response 0x02-0xFF – reserved
startIndex	Integer	0x00-0xff	If the Request type for this command is Extended response, the startIndex provides the starting index for the requested elements of the associated devices list

2987 **2.4.3.1.1.1 When Generated**

2988 The NWK_addr_req is generated from a Local Device wishing to inquire as to the 16-bit address of the
2989 Remote Device based on its known IEEE address. The destination addressing on this command shall be
2990 unicast or broadcast to all devices for which macRxOnWhenIdle = TRUE.

2991 **2.4.3.1.1.2 Effect on Receipt**

2992 Upon receipt, a Remote Device shall compare the IEEEAddr to its *nwkIeeeAddress* in the NIB or any IEEE
2993 address held in its *nwkNeighborTable* where the Device Type field of the entry is 0x02 (End Device). If
2994 there is no match and the request was unicast, a NWK_addr_resp command shall be generated and sent
2995 back to the local device with the Status field set to

2996 DEVICE_NOT_FOUND, the IEEEAddrRemoteDev field set to the IEEE address of the request; the
2997 NWKAddrRemoteDev field set to the NWK address of this device; and the NumAssocDev, StartIndex, and
2998 NWKAddrAssocDevList fields shall not be included in the frame. If there is no match and the command
2999 was received as a broadcast, the request shall be discarded and no response generated.

3000 If a match is detected between the contained IEEEAddr and the receiving device's *nwkIeeeAddress* or one
3001 held in the receiving device's *nwkNeighborTable*, the RequestType shall be used to create a response. If the
3002 RequestType is one of the reserved values, a NWK_addr_resp command shall be generated and sent back
3003 to the local device with the Status field set to INV_REQUESTTYPE; the IEEEAddrRemoteDev field set to
3004 the IEEE address of the request; the NWKAddrRemoteDev field set to the network address corresponding
3005 to the IEEE address in the request; the NumAssocDev, StartIndex, and NWKAddrAssocDevList fields
3006 shall not be included in the frame.

3007 If the RequestType is single device response, a NWK_addr_resp command shall be generated and sent
3008 back to the local device with the Status field set to SUCCESS, the IEEEAddrRemoteDev field set to the
3009 IEEE address of the request; the NWKAddrRemoteDev field set to the NWK address of the discovered
3010 device; and the NumAssocDev, StartIndex, and NWKAddrAssocDevList fields shall not be included in the
3011 frame.

If the RequestType was Extended response and the Remote Device is either the ZigBee coordinator or router, a NWK_addr_resp command shall be generated and sent back to the local device with the Status field set to SUCCESS, the IEEEAddrRemoteDev field set to the IEEE address of the device itself, and the NWKAddrRemoteDev field set to the NWK address of the device itself. The Remote Device shall also supply a list of all 16-bit NWK addresses in the NWKAddrAssocDevList field, starting with the entry StartIndex and continuing with whole entries until the maximum APS packet length is reached, for all devices in its *nwkNeighborTable* where the Device Type is 0x02 (End Device). It shall then set the NumAssocDev field to the number of entries in the NWKAddrAssocDevList field.

2.4.3.1.2 IEEE_addr_req

The IEEE_addr_req command (ClusterID=0x0001) shall be formatted as illustrated in Figure 2.21.

Figure 2.21 Format of the IEEE_addr_req Command Frame

Octets: 2	1	1
NWKAddrOfInterest	RequestType	StartIndex

Table 2.47 specifies the fields of the IEEE_addr_req command frame.

Table 2.47 Fields of the IEEE_addr_req Command

Name	Type	Valid Range	Description
NWKAddrOfInterest	Device Address	16-bit NWK address	NWK address that is used for IEEE address mapping.
RequestType	Integer	0x00-0xff	Request type for this command: 0x00 – Single device response 0x01 – Extended response 0x02-0xff – reserved
startIndex	Integer	0x00-0xff	If the Request type for this command is Extended response, the StartIndex provides the starting index for the requested elements of the associated devices list.

2.4.3.1.2.1 When Generated

The IEEE_addr_req is generated from a Local Device wishing to inquire as to the 64-bit IEEE address of the Remote Device based on their known 16-bit address. The destination addressing on this command shall be unicast.

2.4.3.1.2.2 Effect on Receipt

Upon receipt a Remote Device shall compare the NWKAddrOfInterest to its local *nwkNetworkAddress* value in the NIB, or compare any Network address field held in its *nwkNeighborTable* that also has the Device Type field set to 0x02 (End Device). If there is no match, an IEEE_addr_resp command shall be generated and sent back to the local device with the Status field set to DEVICE_NOT_FOUND; the IEEEAddrRemoteDev field set to the IEEE address of this device; the NWKAddrRemoteDev field set to the NWK address of the request; and the NumAssocDev, StartIndex, and NWKAddrAssocDevList fields shall not be included in the frame.

If a match is detected between the contained NWKAddrOfInterest and the receiving device's *nwkNetworkAddress* or one held in the *nwkNeighborTable*, the RequestType shall be used to create a response. If the RequestType is one of the reserved values, an IEEE_addr_resp command shall be generated and sent back to the local device with the Status field set to INV_REQUESTTYPE, the IEEEAddrRemoteDev field set to the IEEE address of this device, the NWKAddrRemoteDev field set to the network address of this device and the NumAssocDev, StartIndex, and NWKAddrAssocDevList fields shall not be included in the frame.

If the RequestType is single device response, an IEEE_addr_resp command shall be generated and sent back to the local device with the Status field set to SUCCESS, the IEEEAddrRemoteDev field set to the IEEE address of the discovered device, the NWKAddrRemoteDev field set to the NWK address of the request and the NumAssocDev, StartIndex, and NWKAddrAssocDevList fields shall not be included in the frame.

If the RequestType indicates an Extended Response and the Remote Device is the ZigBee coordinator or router with associated devices, an IEEE_addr_resp command shall be generated and sent back to the local device with the Status field set to SUCCESS, the IEEEAddrRemoteDev field set to the IEEE address of the device itself, and the NWKAddrRemoteDev field set to the NWK address of the device itself. The Remote Device shall also supply a list of all 16-bit network addresses in the NWKAddrAssocDevList field, starting with the entry StartIndex and continuing with whole entries until the maximum APS packet length is reached, for each entry in the *nwkNeighborTable* where the Device Type field is set to 0x02 (End Device). It shall then set the NumAssocDev field to the number of entries in the NWKAddrAssocDevList field.

2.4.3.1.3 Node_Desc_req

The Node_Desc_req_command (ClusterID=0x0002) shall be formatted as illustrated in Figure 2.22.

Figure 2.22 Format of the Node_Desc_req Command Frame

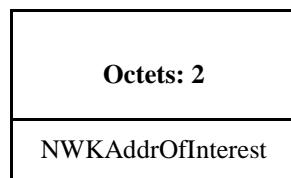


Table 2.48 specifies the fields for the Node_Desc_req command frame.

Table 2.48 Fields of the Node_Desc_req Command

Name	Type	Valid Range	Description
NWKAddrOfInterest	Device Address	16-bit NWK address	NWK address for the request

2.4.3.1.3.1 When Generated

The Node_Desc_req command is generated from a local device wishing to inquire as to the node descriptor of a remote device. This command shall be unicast either to the remote device itself or to an alternative device that contains the discovery information of the remote device.

The local device shall generate the Node_Desc_req command using the format illustrated in Table 2.48. The NWKAddrOfInterest field shall contain the network address of the remote device for which the node descriptor is required.

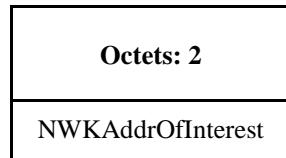
2.4.3.1.3.2 Effect on Receipt

Upon receipt of this command, the recipient device shall process the command and generate a Node_Desc_rsp command in response, according to the description in section 2.4.4.2.3.1.

3075 **2.4.3.1.4 Power_Desc_req**

3076 The Power_Desc_req command (ClusterID=0x0003) shall be formatted as illustrated in Figure 2.23.

3077 **Figure 2.23 Format of the Power_Desc_req Command Frame**



3078

3079 Table 2.49 specifies the fields of the Power_Desc_req command frame.

3080 **Table 2.49 Fields of the Power_Desc_req Command**

Name	Type	Valid Range	Description
NWKAddrOfInterest	Device Address	16-bit NWK address	NWK address for the request.

3081 **2.4.3.1.4.1 When Generated**

3082 The Power_Desc_req command is generated from a local device wishing to inquire as to the power descriptor of a remote device. This command shall be unicast either to the remote device itself or to an alternative device that contains the discovery information of the remote device.

3085 The local device shall generate the Power_Desc_req command using the format illustrated in Table 2.49.
3086 The NWKAddrOfInterest field shall contain the network address of the remote device for which the power
3087 descriptor is required.

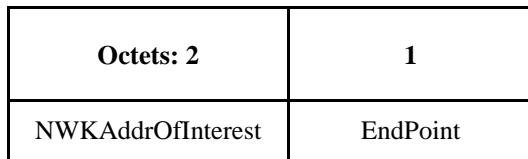
3088 **2.4.3.1.4.2 Effect on Receipt**

3089 Upon receipt of this command, the recipient device shall process the command and generate a Pow-
3090 er_Desc_rsp command in response according to the description in section 2.4.4.2.4.1.

3091 **2.4.3.1.5 Simple_Desc_req**

3092 The Simple_Desc_req command (ClusterID=0x0004) shall be formatted as illustrated in Figure 2.24.

3093 **Figure 2.24 Format of the Simple_Desc_req Command Frame**



3094

3095 Table 2.50 specifies the fields of the Simple_Desc_req command frame.

3096

Table 2.50 Fields of the Simple_Desc_req Command

Name	Type	Valid Range	Description
NWKAddrOfInterest	Device Address	16-bit NWK address	NWK address for the request
Endpoint	8 bits	1–254	The endpoint on the destination

3097

2.4.3.1.5.1 When Generated

3098
3099
3100

The Simple_Desc_req command is generated from a local device wishing to inquire as to the simple descriptor of a remote device on a specified endpoint. This command shall be unicast either to the remote device itself or to an alternative device that contains the discovery information of the remote device.

3101
3102
3103
3104

The local device shall generate the Simple_Desc_req command using the format illustrated in Table 2.50. The NWKAddrOfInterest field shall contain the network address of the remote device for which the simple descriptor is required and the endpoint field shall contain the endpoint identifier from which to obtain the required simple descriptor.

3105

2.4.3.1.5.2 Effect on Receipt

3106
3107

Upon receipt of this command, the recipient device shall process the command and generate a Simple_Desc_rsp command in response, according to the description in section 2.4.4.2.5.1.

3108

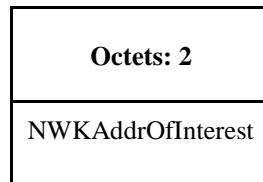
2.4.3.1.6 Active_EP_req

3109

The Active_EP_req command (ClusterID=0x0005) shall be formatted as illustrated in Figure 2.25.

3110

Figure 2.25 Format of the Active_EP_req Command Frame



3111

3112

Table 2.51 specifies the fields of the Active_EP_req command frame.

3113

Table 2.51 Fields of the Active_EP_req Command

Name	Type	Valid Range	Description
NWKAddrOfInterest	Device Address	16-bit NWK address	NWK address for the request.

3114

2.4.3.1.6.1 When Generated

3115
3116
3117

The Active_EP_req command is generated from a local device wishing to acquire the list of endpoints on a remote device with simple descriptors. This command shall be unicast either to the remote device itself or to an alternative device that contains the discovery information of the remote device.

3118
3119
3120

The local device shall generate the Active_EP_req command using the format illustrated in Table 2.51. The NWKAddrOfInterest field shall contain the network address of the remote device for which the active endpoint list is required.

3121 **2.4.3.1.6.2 Effect on Receipt**

3122 Upon receipt of this command, the recipient device shall process the command and generate an Active_EP_rsp command in response, according to the description in section 2.4.4.2.6.1.

3124 **2.4.3.1.7 Match_Desc_req**

3125 The Match_Desc_req command (ClusterID=0x0006) shall be formatted as illustrated in Figure 2.26.

3126 **Figure 2.26 Format of the Match_Desc_req Command Frame**

Octets: 2	2	1	Variable	1	Variable
NWKAddrOfInterest	ProfileID	NumInClusters	InClusterList	NumOutClusters	OutClusterList

3127

3128 Table 2.52 specifies the fields of the Match_Desc_req command frame.

3129 **Table 2.52 Fields of the Match_Desc_req Command**

Name	Type	Valid Range	Description
NWKAddrOfInterest	Device Address	16-bit NWK address	NWK address for the request.
ProfileID	Integer	0x0000-0xffff	Profile ID to be matched at the destination.
NumInClusters	Integer	0x00-0xff	The number of Input Clusters provided for matching within the InClusterList.
InClusterList	2 bytes * NumInClusters		List of Input ClusterIDs to be used for matching; the InClusterList is the desired list to be matched by the Remote Device (the elements of the InClusterList are the supported output clusters of the Local Device).
NumOutClusters	Integer	0x00-0xff	The number of Output Clusters provided for matching within OutClusterList.
OutClusterList	2 bytes * NumOutClusters		List of Output ClusterIDs to be used for matching; the OutClusterList is the desired list to be matched by the Remote Device (the elements of the OutClusterList are the supported input clusters of the Local Device).

3130 **2.4.3.1.7.1 When Generated**

3131 The Match_Desc_req command is generated from a local device wishing to find remote devices supporting
3132 a specific simple descriptor match criterion. This command shall either be broadcast to all devices for
3133 which macRxOnWhenIdle = TRUE, or unicast. If the command is unicast, it shall be directed either to the
3134 remote device itself or to an alternative device that contains the discovery information of the remote device.

3135 The local device shall generate the Match_Desc_req command using the format illustrated in Table 2.52.
3136 The NWKAddrOfInterest field shall contain the network address indicating a broadcast to all devices for
3137 which macRxOnWhenIdle = TRUE (0xffffd) if the command is to be broadcast, or the network address of
3138 the remote device for which the match is required.

3139 The remaining fields shall contain the required criterion for which the simple descriptor match is requested.
3140 The ProfileID field shall contain the identifier of the profile for which the match is being sought or the
3141 wildcard profile ID of 0xFFFF.

3142 The NumInClusters field shall contain the number of elements in the InClusterList field. If the value of this
3143 field is 0, the InClusterList field shall not be included. If the value of the NumInClusters field is not equal
3144 to 0, the InClusterList field shall contain the list of input cluster identifiers for which the match is being
3145 sought.

3146 The NumOutClusters field shall contain the number of elements in the OutClusterList field. If the value of
3147 this field is 0, the OutClusterList field shall not be included. If the value of the NumOutClusters field is not
3148 equal to 0, the OutClusterList field shall contain the list of output cluster identifiers for which the match is
3149 being sought.

3150 **2.4.3.1.7.2 Effect on Receipt**

3151 Upon receipt of this command, the recipient device shall process the command and generate a
3152 Match_Desc_rsp command in response, according to the description in section 2.4.4.2.7.1.

3153 **2.4.3.1.8 Complex_Desc_req**

3154 The Complex_Desc_req command (ClusterID=0x0010) shall be formatted as illustrated in Figure 2.27.

3155 **Figure 2.27 Format of the Complex_Desc_req Command Frame**



3156 Table 2.53 specifies the fields of the Complex_Desc_req command frame.

3157 **Table 2.53 Fields of the Complex_Desc_req Command**

Name	Type	Valid Range	Description
NWKAddrOfInterest	Device Address	16-bit NWK address	NWK address for the request

3159 **2.4.3.1.8.1 When Generated**

3160 The Complex_Desc_req command is generated from a local device wishing to inquire as to the complex
3161 descriptor of a remote device. This command shall be unicast either to the remote device itself or to an alter-
3162 native device that contains the discovery information of the remote device.

3163 The local device shall generate the Complex_Desc_req command using the format illustrated in Table 2.53.
3164 The NWKAddrOfInterest field shall contain the network address of the remote device for which the com-
3165 plex descriptor is required.

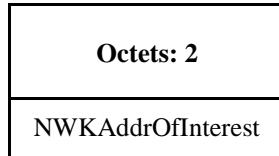
3166 **2.4.3.1.8.2 Effect on Receipt**

3167 Upon receipt of this command, the recipient device shall process the command and generate a Com-
3168 plex_Desc_rsp command in response, according to the description in section 2.4.4.2.8.1.

3169 **2.4.3.1.9 User_Desc_req**

3170 The User_Desc_req (ClusterID=0x0011) command shall be formatted as illustrated in Figure 2.28.

3171 **Figure 2.28 Format of the User_Desc_req Command Frame**



3172

3173 Table 2.54 specifies the fields of the User_Desc_req command frame.

3174 **Table 2.54 Fields of the User_Desc_req Command**

Name	Type	Valid Range	Description
NWKAddrOfInterest	Device Address	16-bit NWK address	NWK address for the request.

3175 **2.4.3.1.9.1 When Generated**

3176 The User_Desc_req command is generated from a local device wishing to inquire as to the user descriptor
3177 of a remote device. This command shall be unicast either to the remote device itself or to an alternative de-
3178 vice that contains the discovery information of the remote device.

3179 The local device shall generate the User_Desc_req command using the format illustrated in Table 2.54. The
3180 NWKAddrOfInterest field shall contain the network address of the remote device for which the user de-
3181 scriptor is required.

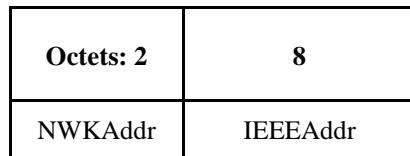
3182 **2.4.3.1.9.2 Effect on Receipt**

3183 Upon receipt of this command, the recipient device shall process the command and generate a Us-
3184 er_Desc_rsp command in response, according to the description in section 2.4.4.2.9.1.

3185 **2.4.3.1.10 Discovery_Cache_req**

3186 The Discovery_Cache_req command (ClusterID=0x0012) shall be formatted as illustrated in Figure 2.29.

3187 **Figure 2.29 Format of the Discovery_Cache_req Command Frame**



3188

3189 Table 2.55 specifies the parameters for the Discovery_Cache_req command frame.

3190 **Table 2.55 Fields of the Discovery_Cache_req Command**

Name	Type	Valid Range	Description
NWKAddr	Device Address	16-bit NWK address	NWK address for the Local Device.
IEEEAddr	Device Address	64-bit IEEE address	IEEE address for the Local Device.

3191 **2.4.3.1.10.1 When Generated**

3192 The Discovery_Cache_req is provided to enable devices on the network to locate a Primary Discovery
3193 Cache device on the network. The destination addressing on this primitive shall be broadcast to all devices
3194 for which macRxOnWhenIdle = TRUE.

3195 **2.4.3.1.10.2 Effect on Receipt**

3196 Upon receipt, if the Remote Device does not support the Discovery_Cache_req, the request shall be
3197 dropped and no further processing performed. If the Discovery_Cache_req is supported, the Remote Device
3198 shall create a unicast Discovery_Cache_rsp message to the source indicated by the Discovery_Cache_req
3199 and include a SUCCESS status.

3200 **2.4.3.1.11 Device_ance**

3201 The Device_ance command (ClusterID=0x0013) shall be formatted as illustrated in Figure 2.30.

3202 **Figure 2.30 Format of the Device_ance Command Frame**

Octets: 2	8	1
NWKAddr	IEEEAddr	Capability

3203

3204 Table 2.56 specifies the fields of the Device_ance command frame.

3205 **Table 2.56 Fields of the Device_ance Command**

Name	Type	Valid Range	Description
NWKAddr	Device Address	16-bit NWK address	NWK address for the Local Device
IEEEAddr	Device Address	64-bit IEEE address	IEEE address for the Local Device
Capability	Bitmap	See Figure 2.17	Capability of the local device

3206 **2.4.3.1.11.1 When Generated**

3207 The Device_ance is provided to enable ZigBee devices on the network to notify other ZigBee devices that
3208 the device has joined or re-joined the network, identifying the device's 64-bit IEEE address and new 16-bit
3209 NWK address, and informing the Remote Devices of the capability of the ZigBee device. This command
3210 shall be invoked for all ZigBee end devices upon join or rejoin. This command may also be invoked by
3211 ZigBee routers upon join or rejoin as part of NWK address conflict resolution. The destination addressing
3212 on this primitive is broadcast to all devices for which macRxOnWhenIdle = TRUE.

3213 **2.4.3.1.11.2 Effect on Receipt**

3214 Upon receipt, the Remote Device shall use the IEEEAddr in the message to find a match with any other
3215 IEEE address held in the Remote Device. If a match is detected, the Remote Device shall update the
3216 nwkAddressMap attribute of the NIB with the updated NWKAddr corresponding to the IEEEAddr re-
3217 ceived.

3218 The Remote Device shall also use the NWKAddr in the message to find a match with any other 16-bit
3219 NWK address held in the Remote Device, even if the IEEEAddr field in the message carries the value of
3220 0xffffffffffff. If a match is detected for a device with an IEEE address other than that indicated in the
3221 IEEEAddr field received, then this entry shall be marked as not having a known valid 16-bit NWK address.

3222 **2.4.3.1.12 Parent_ance**

3223 The Parent_ance command (ClusterID = 0x001F) shall be formatted as illustrated in Figure 2.31.

3224
3225

Figure 2.31 Format of the Parent Anne Message

Octets: 1	Variable	...	Variable
NumberOfChildren	ChildInfo[0]	...	ChildInfo[n]

3226
3227
3228
3229

Table 2.57 specifies the contents of the ChildInfo structure.

Table 2.57 - Format of the ChildInfo Structure

Name	Type	Description
Extended Address	64-bit IEEE address	The IEEE address of the child bound to the parent.

3230
3231

2.4.3.1.12.1 When Generated

3232
3233
3234
3235
3236
3237

The Parent_ance is provided to enable ZigBee routers (including the coordinator) on the network to notify other ZigBee routers about all the end devices known to the local device. This command provides a means to resolve conflicts more quickly than aging out the child, when multiple routers purport to be the active parent of a particular end-device. The command may be broadcast from one router to all routers and the coordinator using the broadcast address 0xFFFF or unicast from one router to another router.

3238
3239
3240
3241
3242
3243
3244
3245

This message must be generated if all the following conditions are met:

1. The router or coordinator device has rebooted.
2. The router or coordinator is operating in the joined and authenticated state.

The message generated under the above circumstances must be broadcast. Before broadcasting a Parent_ance message, the device shall start a countdown timer, *apsParentAnnounceTimer* equal to *apsParentAnnounceBaseTimer* + a random value from 0 to *apsParentAnnounceJitterMax*.

3246 When the timer expires, a router shall examine its neighbor table for all devices. The router shall construct, but not yet send, an empty Parent_ance message and set NumberOfChildren to 0. For each end device in the neighbor table, it shall do the following.

- 3249 1. If the Neighbor Table entry indicates a Device Type not equal to End Device (0x02), do not process this entry. Continue to the next one.
- 3250 2. Incorporate end device information into the Parent_ance message by doing the following:
- 3251 a. Append a ChildInfo structure to the message.
- 3252 b. Increment NumberOfChildren by 1.
- 3253 3. Note: The value of Keepalive Received for the Neighbor Table Entry is not considered.

3255 After processing all entries in the neighbor table, if the NumberOfChildren is greater than 0, then it shall send the message to the all routers broadcast address (0xFFFF). If NumberOfChildren is 0, it shall discard the previously constructed Parent_ance message and not send it.

3258 If the device has more ChildInfo entries than fit in a single message, it shall send additional messages. Each additional message needed shall trigger the device to calculate and start a new apsParentAnnounceTimer equal to apsParentAnnounceBaseTimer + a random value from 0 to apsParentAnnounceJitterMax. The local device shall wait until that timer expires before sending each additional message. . The NumberOfChildren for each message shall be set according to the number of ChildInfo entries contained within the message.

3264 If the device must send multiple Parent_ance message but receives a keepalive from an end device before it has sent the Parent_Ance message, it shall not include that device in the message.

3266

2.4.3.1.12.2 Effect on receipt

3268 If the message is received by an end device, it shall be dropped. No further processing shall be done.

3269 Upon receipt of a broadcast Parent_ance, if the local device has a non-zero value for its apsParentAnnounceTimer it shall immediately re-calculate a new value and start a new countdown. The apsParentAnnounceTimer shall be set to apsParentAnnounceBaseTimer + a random value from 0 to apsParentAnnounceJitterMax. It shall continue processing the message.

3273 A router shall construct, but not yet send, an empty Parent_Ance_Rsp message with NumberOfChildren set to 0. It shall examine each Extended Address present in the message and search its Neighbor Table for an Extended Address entry that matches. For each match, process as follows:

- 3276 1. If the Device Type is Zigbee End Device (0x02) and the Keepalive Received value is TRUE, do the following:
- 3277 a. It shall append to the Parent_ance_rsp frame the ChildInfo structure.
- 3278 b. Increment the NumberOfChildren by 1.
- 3279 2. If the Device Type is not ZigBee End Device (0x02) or the Keepalive Received value is FALSE, do not process any further. Continue to the next entry.

3282

3283 If the NumberOfChildren field value is 0, the local device shall discard the previously constructed Parent_Ance_rsp. No response message shall be sent.

3285 If the NumberOfChildren field in the Parent_Ance_rsp is greater than 0, it shall unicast the message to the sender of the Parent_Ance message.

3287 If the device has more ChildInfo entries than fit in a single message, it shall send additional messages. These messages do not have to be jittered or delayed since they are unicast to a single device. Each Parent_ance_rsp shall set the NumberOfChildren field to the number of entries contained within the message.

3290

2.4.3.1.13 User_Desc_set

3292 The User_Desc_set command (ClusterID=0x0014) shall be formatted as illustrated in Figure 2.32.

3293 **Figure 2.32 Format of the User_Desc_set Command Frame**

Octets: 2	1	Various
NWKAddrOfInterest	Length	UserDescriptor

3294

3295 Table 2.58 specifies the fields of the User_Desc_set command frame.

3296 **Table 2.58 Fields of the User_Desc_set Command**

Name	Type	Valid Range	Description
NWKAddrOfInterest	Device Address	16-bit NWK address	NWK address for the request.
Length	Integer	0x00 - 0x10	Length of the User Descriptor in bytes.
UserDescription	User Descriptor		The user description to configure; if the ASCII character string to be entered here is less than 16 characters in length, it shall be padded with space characters (0x20) to make a total length of 16 characters. Characters with codes 0x00-0x1f are not permitted.

3297 **2.4.3.1.13.1 When Generated**

3298 The User_Desc_set command is generated from a local device wishing to configure the user descriptor on a
3299 remote device. This command shall be unicast either to the remote device itself or to an alternative device
3300 that contains the discovery information of the remote device.

3301 The local device shall generate the User_Desc_set command using the format illustrated in Table 2.58. The
3302 NWKAddrOfInterest field shall contain the network address of the remote device for which the user
3303 descriptor is to be configured and the UserDescription field shall contain the ASCII character string that is to
3304 be configured in the user descriptor. Characters with ASCII codes numbered 0x00 through 0x1f are not
3305 permitted to be included in this string.

3306 **2.4.3.1.13.2 Effect on Receipt**

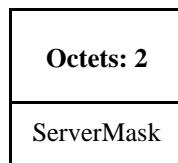
3307 Upon receipt of this command, the recipient device shall process the command and generate a User_Desc_conf
3308 command in response, according to the description in section 2.4.4.2.11.1.

3309 **2.4.3.1.14 System_Server_Discovery_req**

3310 The System_Server_Discovery_req command (ClusterID=0x0015) shall be formatted as illustrated in Fig-
3311 ure 2.33.

3312

Figure 2.33 Format of the System_Server_Discovery_req Command Frame



3313

3314 Table 2.59 specifies the fields of the System_Server_Discovery_req command frame.

Table 2.59 Fields of the System_Server_Discovery_req Command

Name	Type	Valid Range	Description
ServerMask	Bitmap	16 bits	See Table 2.32 for bit assignments

3316 **2.4.3.1.14.1 When Generated**

3317 The System_Server_Discovery_req is generated from a Local Device wishing to discover the location of a
3318 particular system server or servers as indicated by the ServerMask parameter. The destination addressing
3319 on this request is ‘broadcast to all devices for which macRxOnWhenIdle = TRUE.’

3320 **2.4.3.1.14.2 Effect on Receipt**

3321 Upon receipt, remote devices shall compare the ServerMask parameter to the Server Mask field in their
3322 own Node descriptor. If no bits are found to match, no action is taken. If any matching bits are found, the
3323 remote device shall send a System_Server_Discovery_rsp back to the originator using unicast transmission
3324 (with acknowledgement request) and indicating the matching bits.

3325 **2.4.3.1.15 Discovery_store_req**

3326 The Discovery_Store_req command (ClusterID=0x0016) shall be formatted as illustrated in Figure 2.34.

Figure 2.34 Format of the Discovery_Store_req Command Frame

Octets: 2	8	1	1	1	1	Variable
NWKAddr	IEEEAddr	NodeDescSize	PowerDescSize	ActiveEPSize	Simple DescCount	Simple DescSizeList

3328

3329 Table 2.60 specifies the fields of the Discovery_store_req command frame.

Table 2.60 Fields of the Discovery_store_req Command

Name	Type	Valid Range	Description
NWKAddr	Device Address	16-bit NWK Address	NWK Address for the Local Device.
IEEEAddr	Device Address	64-bit IEEE Address	IEEE Address for the Local Device.

Name	Type	Valid Range	Description
NodeDescSize	Integer	0x00-0xff	Size in bytes of the Node Descriptor for the Local Device.
PowerDescSize	Integer	0x00 - 0xff	Size in bytes of the Power Descriptor for the Local Device.
ActiveEPSIZE	Integer	0x00 - 0xff	Size in bytes of the ActiveEPCount and ActiveEPList fields of the Active_EP_rsp for the Local Device.
SimpleDescCount	Integer	0x00 - 0xff	Number of Simple Descriptors supported by the Local Device (should be the same value as the ActiveEPSIZE).
SimpleDescSizeList	Array of bytes		List of bytes of SimpleDescCount length, each of which represents the size in bytes of the Simple Descriptor for each Active Endpoint on the Local Device.

3331 **2.4.3.1.15.1 When Generated**

3332 The Discovery_store_req is provided to enable ZigBee end devices on the network to request storage of
3333 their discovery cache information on a Primary Discovery Cache device. Included in the request is the
3334 amount of storage space the Local Device requires.

3335 The destination addressing on this request is unicast.

3336 **2.4.3.1.15.2 Effect on Receipt**

3337 Upon receipt, the Remote Device shall determine whether it is a Primary Discovery Cache device. If it is
3338 not a Primary Discovery Cache device, the Remote Device shall return a status of NOT_SUPPORTED.
3339 Next, the Remote Device shall determine whether it has storage for the requested discovery cache size de-
3340 termined by summing the sizes of the NWKAddr and IEEEAddr plus the NodeDescSize, PowerDescSize,
3341 ActiveEPSIZE, and the sizes from the SimpleDescSizeList. If sufficient space exists, the Local Device shall
3342 be provided a SUCCESS status. Otherwise, the Local Device shall return INSUFFICIENT_SPACE. If a
3343 SUCCESS status is returned, the Remote Device shall reserve the storage requested for the upload of the
3344 discovery information from the Local Device. Additionally, if the Local Device supplies an IEEEAddr
3345 which matches a previously stored entry, but the NWKAddr differs from the previous entry, the Remote
3346 Device shall remove the previous entry and discovery cache information in favor of the newly registered
3347 data.

3348 **2.4.3.1.16 Node_Desc_store_req**

3349 The Node_Desc_store_req command (ClusterID=0x0017) shall be formatted as illustrated in Figure 2.35.

3350 **Figure 2.35 Format of the Node_Desc_store_req Command Frame**

Octets: 2	8	Variable
NWKAddr	IEEEAddr	NodeDescriptor

3352 Table 2.61 specifies the fields of the Node_Desc_store_req command frame.

3353 **Table 2.61 Fields of the Node_Desc_store_req Command**

Name	Type	Valid Range	Description
NWKAddr	Device Address	16-bit NWK Address	NWK Address for the Local Device
IEEEAddr	Device Address	64-bit IEEE Address	IEEE address for the Local Device
NodeDescriptor	Node Descriptor		See the Node Descriptor format in section 2.3.2.3

3354 **2.4.3.1.16.1 When Generated**

3355 The Node_Desc_store_req is provided to enable ZigBee end devices on the network to request storage of
3356 their Node Descriptor on a Primary Discovery Cache device which has previously received a SUCCESS
3357 status from a Discovery_store_req to the same Primary Discovery Cache device. Included in this request is
3358 the Node Descriptor the Local Device wishes to cache.

3359 **2.4.3.1.16.2 Effect on Receipt**

3360 Upon receipt, the Remote Device shall determine whether it is a Primary Discovery Cache device. If it is
3361 not a Primary Discovery Cache device, the Remote Device shall return a status of NOT_SUPPORTED.
3362 Next, the Remote Device shall determine whether it has previously processed a Discovery_store_req for the
3363 Local Device and returned a status of SUCCESS. If a previous Discovery_store_req has not been processed
3364 with a SUCCESS status, the Remote Device shall return NOT_PERMITTED. Next, the Remote Device
3365 shall determine if enough space is available to store the Node Descriptor for the Local Device. If not, the
3366 Remote Device shall return INSUFFICIENT_SPACE. Finally, the Remote Device shall store the Node
3367 Descriptor for the Local Device and return SUCCESS. If the request returned a status of SUCCESS and the
3368 NWKAddr and IEEEAddr in the request referred to addresses already held in the Primary Discovery
3369 Cache, the descriptor in this request shall overwrite the previously held entry.

3370 **2.4.3.1.17 Power_Desc_store_req**

3371 The Power_Desc_store_req command (ClusterID=0x0018) shall be formatted as illustrated in Figure 2.36.

3372 **Figure 2.36 Format of the Power_Desc_store_req Command Frame**

Octets: 2	8	Variable
NWKAddr	IEEEAddr	PowerDescriptor

3373

3374 Table 2.62 specifies the fields of the Power_Desc_store_req command frame.

3375 **Table 2.62 Fields of the Power_Desc_store_req Command**

Name	Type	Valid Range	Description
NWKAddr	Device Address	16-bit NWK Address	NWK Address for the Local Device.
IEEEAddr	Device Address	64-bit Address	IEEE address for the Local Device.

Name	Type	Valid Range	Description
PowerDescriptor	Power Descriptor		See the Power Descriptor format in section 2.3.2.4; This field shall only be included in the frame if the status field is equal to SUCCESS.

3376 **2.4.3.1.17.1 When Generated**

3377 The Power_Desc_store_req is provided to enable ZigBee end devices on the network to request storage of
 3378 their Power Descriptor on a Primary Discovery Cache device which has previously received a SUCCESS
 3379 status from a Discovery_store_req to the same Primary Discovery Cache device. Included in this request is
 3380 the Power Descriptor the Local Device wishes to cache.

3381 **2.4.3.1.17.2 Effect on Receipt**

3382 Upon receipt, the Remote Device shall determine whether it is a Primary Discovery Cache device. If it is
 3383 not a Primary Discovery Cache device, the Remote Device shall return a status of NOT_SUPPORTED.
 3384 Next, the Remote Device shall determine whether it has previously processed a Discovery_store_req for the
 3385 Local Device and returned a status of SUCCESS. If a previous Discovery_store_req has not been processed
 3386 with a SUCCESS status, the Remote Device shall return NOT_PERMITTED. Next, the Remote Device
 3387 shall determine if enough space is available to store the Power Descriptor for the Local Device. If not, the
 3388 Remote Device shall return INSUFFICIENT_SPACE. Finally, the Remote Device shall store the Power
 3389 Descriptor for the Local Device and return SUCCESS. If the request returned a status of SUCCESS, and
 3390 the NWKAddr and IEEEAddr in the request referred to addresses already held in the Primary Discovery
 3391 Cache, the descriptor in this request shall overwrite the previously held entry.

3392 **2.4.3.1.18 Active_EP_store_req**

3393 The Active_EP_store_req command (ClusterID=0x0019) shall be formatted as illustrated in Figure 2.37.

3394 **Figure 2.37 Format of the Active_EP_store_req Command Frame**

Octets: 2	8	1	Variable
NWKAddr	IEEEAddr	ActiveEPCount	ActiveEPList

3395

3396 Table 2.63 specifies the fields of the Active_EP_store_req command frame.

3397 **Table 2.63 Fields of the Active_EP_store_req Command**

Name	Type	Valid Range	Description
NWKAddr	Device Address	16-bit NWK Address	NWK Address for the Local Device.
IEEEAddr	Device Address	64-bit IEEE Address	IEEE Address for the Local Device.
ActiveEPCount	Integer	0x00-0xff	The count of active endpoints on the Local Device.

Name	Type	Valid Range	Description
ActiveEPLList			List of bytes, each of which represents an 8-bit endpoint number.

3398 **2.4.3.1.18.1 When Generated**

3399 The Active_EP_store_req is provided to enable ZigBee end devices on the network to request storage of
 3400 their list of Active Endpoints on a Primary Discovery Cache device which has previously received a
 3401 SUCCESS status from a Discovery_store_req to the same Primary Discovery Cache device. Included in
 3402 this request is the count of Active Endpoints the Local Device wishes to cache and the endpoint list itself.

3403 **2.4.3.1.18.2 Effect on Receipt**

3404 Upon receipt, the Remote Device shall determine whether it is a Primary Discovery Cache device. If it is
 3405 not a Primary Discovery Cache device, the Remote Device shall return a status of NOT_SUPPORTED.
 3406 Next, the Remote Device shall determine whether it has previously processed a Discovery_store_req for the
 3407 Local Device and returned a status of SUCCESS. If a previous Discovery_store_req has not been processed
 3408 with a SUCCESS status, the Remote Device shall return NOT_PERMITTED. Next, the Remote Device
 3409 shall determine if enough space is available to store the Active Endpoint count and list for the Local De-
 3410 vice. If not, the Remote Device shall return INSUFFICIENT_SPACE. Finally, the Remote Device shall
 3411 store the Active Endpoint count and list for the Local Device and return SUCCESS. If the request returned
 3412 a status of
 3413 SUCCESS, and the NWKAddr and the IEEEAddr in the request referred to addresses already held in the
 3414 Primary Discovery Cache, the descriptor in this request shall overwrite the previously held entry.

3415 **2.4.3.1.19 Simple_Desc_store_req**

3416 The Simple_Desc_store_req command (ClusterID=0x001a) shall be formatted as illustrated in Figure 2.38.

3417 **Figure 2.38 Format of the Simple_Desc_store_req Command Frame**

Octets: 2	8	1	Variable
NWKAddr	IEEEAddr	Length	SimpleDescriptor

3418

3419 Table 2.64 specifies the fields of the Simple_Desc_store_req command frame.

3420 **Table 2.64 Fields of the Simple_Desc_store_req Command**

Name	Type	Valid Range	Description
NWKAddr	Device Address	16-bit NWK Address	NWK Address for the Local Device.
IEEEAddr	Device Address	64-bit IEEE Address	IEEE Address for the Local Device.
Length	Device Address	0x00 - 0xff	The length in bytes of the Simple Descriptor to follow.

Name	Type	Valid Range	Description
SimpleDescriptor	Simple Descriptor		See the Simple Descriptor format in section 2.3.2.5.

3421 **2.4.3.1.19.1 When Generated**

3422 The Simple_desc_store_req is provided to enable ZigBee end devices on the network to request storage of
 3423 their list of Simple Descriptors on a Primary Discovery Cache device which has previously received a
 3424 SUCCESS status from a Discovery_store_req to the same Primary Discovery Cache device. Note that each
 3425 Simple Descriptor for every active endpoint on the Local Device must be individually uploaded to the Pri-
 3426 mary Discovery Cache device via this command to enable cached discovery. Included in this request is the
 3427 length of the Simple Descriptor the Local Device wishes to cache and the Simple Descriptor itself. The
 3428 endpoint is a field within the Simple Descriptor and is accessed by the Remote Device to manage the dis-
 3429 covery cache information for the Local Device.

3430 **2.4.3.1.19.2 Effect on Receipt**

3431 Upon receipt, the Remote Device shall determine whether it is a Primary Discovery Cache device. If it is
 3432 not a Primary Discovery Cache device, the Remote Device shall return a status of NOT_SUPPORTED.
 3433 Next, the Remote Device shall determine whether it has previously processed a Discovery_store_req for the
 3434 Local Device and returned a status of SUCCESS. If a previous Discovery_store_req has not been processed
 3435 with a SUCCESS status, the Remote Device shall return NOT_PERMITTED. Next, the Remote Device
 3436 shall determine if enough space is available to store the Simple Descriptor for the Local Device. If not, the
 3437 Remote Device shall return INSUFFICIENT_SPACE. Finally, the Remote Device shall store the Simple
 3438 Descriptor for the Local Device and return SUCCESS. If the request returned a status of SUCCESS and the
 3439 NWKAddr and the IEEEAddr in the request referred to addresses already held in the Primary Discovery
 3440 Cache, the descriptor in this request shall overwrite the previously held entry.

3441 **2.4.3.1.20 Remove_node_cache_req**

3442 The Remove_node_cache_req command (ClusterID=0x001b) shall be formatted as illustrated in Figure
 3443 2.39.

3444 **Figure 2.39 Format of the Remove_node_cache_req Command Frame**

Octets: 2	8
NWKAddr	IEEEAddr

3445

3446 Table 2.65 specifies the fields of the Remove_node_cache_req command frame.

3447 **Table 2.65 Fields of the Remove_node_cache_req Command**

Name	Type	Valid Range	Description
NWKAddr	Device Address	16-bit NWK Address	NWK Address for the device of interest.
IEEEAddr	Device Address	64-bit IEEE Address	IEEE Address for the device of interest.

3448 **2.4.3.1.20.1 When Generated**

3449 The Remove_node_cache_req is provided to enable ZigBee devices on the network to request removal of
3450 discovery cache information for a specified ZigBee end device from a Primary Discovery Cache device.
3451 The effect of a successful Remove_node_cache_req is to undo a previously successful Discovery_store_req
3452 and additionally remove any cache information stored on behalf of the specified ZigBee end device on the
3453 Primary Discovery Cache device.

3454 **2.4.3.1.20.2 Effect on Receipt**

3455 Upon receipt, the Remote Device shall determine whether it is a Primary Discovery Cache device. If it is
3456 not a Primary Discovery Cache device, the Remote Device shall return a status of NOT_SUPPORTED.
3457 Next, the Remote Device shall determine whether it has previously processed a Discovery_store_req for the
3458 indicated device and returned a status of SUCCESS. If a previous Discovery_store_req has not been pro-
3459 cessed with a SUCCESS status, the Remote Device shall return DEVICE_NOT_FOUND. Finally, the Re-
3460 mote Device shall remove all cached discovery information for the device of interest and return SUCCESS
3461 to the Local Device.

3462 **2.4.3.1.21 Find_node_cache_req**

3463 The Find_node_cache_req command (ClusterID=0x001c) shall be formatted as illustrated in Figure 2.40.

3464 **Figure 2.40 Format of the Find_node_cache Command Frame**

Octets: 2	8
NWKAddr	IEEEAddr

3465

3466 Table 2.66 specifies the fields of the Find_node_cache_req command frame.

3467 **Table 2.66 Fields of the Find_node_cache_req Command Frame**

Name	Type	Valid Range	Description
NWKAddr	Device Address	16-bit NWK Address	NWK Address for the device of interest.
IEEEAddr	Device Address	64-bit IEEE Address	IEEE Address for the device of interest.

3468 **2.4.3.1.21.1 When Generated**

3469 The Find_node_cache_req is provided to enable ZigBee devices on the network to broadcast to all devices
3470 for which macRxOnWhenIdle = TRUE a request to find a device on the network that holds discovery in-
3471 formation for the device of interest, as specified in the request parameters. The effect of a successful
3472 Find_node_cache_req is to have the Primary Discovery Cache device, holding discovery information for
3473 the device of interest, unicast a Find_node_cache_rsp back to the Local Device. Note that, like the
3474 NWK_addr_req, only the device meeting this criteria shall respond to the request generated by
3475 Find_node_cache_req.

3476 **2.4.3.1.21.2 Effect on Receipt**

3477 Upon receipt, the Remote Device shall determine whether it is the device of interest or a Primary Discovery
3478 Cache device, and if so, if it holds discovery cache information for the device of interest. If it is not the de-
3479 vice of interest or a Primary Discovery Cache device, and does not hold discovery cache information for
3480 the device of interest, the Remote Device shall cease processing the request and not supply a response. If
3481 the Remote Device is the device of interest, or a Primary Discovery Cache device, and, if the device holds
3482 discovery information for the indicated device of interest, the Remote Device shall return the NWKAddr
3483 and IEEEaddr for the device of interest.

3484 **2.4.3.1.22 Extended_Simple_Desc_req**

3485 The Extended_Simple_Desc_req command (ClusterID=0x001d) shall be formatted as illustrated in Figure
3486 2.41.

3487 **Figure 2.41 Format of the Extended_Simple_Desc_req Command Frame**

Octets: 2	1	1
NWKAddrOfInterest	EndPoint	StartIndex

3488

3489 Table 2.67 specifies the fields of the Extended_Simple_Desc_req command frame.

3490 **Table 2.67 Fields of the Extended_Simple_Desc_req Command**

Name	Type	Valid Range	Description
NWKAddrOfInterest	Device Address	16-bit NWK address	NWK address for the request.
Endpoint	8 bits	1-254	The endpoint on the destination.
StartIndex	8 bits	0x00-0xff	Starting index within the cluster list of the response represented by an ordered list of the Application Input Cluster List and Application Output Cluster List.

3491 **2.4.3.1.22.1 When Generated**

3492 The Extended_Simple_Desc_req command is generated from a local device wishing to inquire as to the
3493 simple descriptor of a remote device on a specified endpoint. This command shall be unicast either to the
3494 remote device itself or to an alternative device that contains the discovery information of the remote device.
3495 The Extended_Simple_Desc_req is intended for use with devices which employ a larger number of appli-
3496 cation input or output clusters than can be described by the Simple_Desc_req.

3497 The local device shall generate the Extended_Simple_Desc_req command using the format illustrated in
3498 Table 2.67. The NWKAddrOfInterest field shall contain the network address of the remote device for
3499 which the simple descriptor is required and the endpoint field shall contain the endpoint identifier from
3500 which to obtain the required simple descriptor. The StartIndex is the first entry requested in the Application
3501 Input Cluster List and Application Output Cluster List sequence within the resulting response.

3502 **2.4.3.1.22.2 Effect on Receipt**

3503 Upon receipt of this command, the recipient device shall process the command and generate an Extend-
3504 ed_Simple_Desc_rsp command in response, according to the description in section 2.4.4.2.20.1.

3505 The results in the Extended_Simple_Desc_rsp shall include the elements described in Table 2.111 with a
3506 selectable set of the application input cluster and application output cluster lists starting with the entry
3507 StartIndex and continuing with whole entries until the maximum APS packet length is reached, along with
3508 a status of SUCCESS.

3509 **2.4.3.1.23 Extended_Active_EP_req**

3510 The Extended_Active_EP_req command (ClusterID=0x001e) shall be formatted as illustrated in Figure
3511 2.42.

3512 **Figure 2.42 Format of the Extended_Active_EP_req Command Frame**

Octets: 2	1
NWKAddrOfInterest	startIndex

3513 Table 2.68 specifies the fields of the Extended_Active_EP_req command frame.
3514

3515 **Table 2.68 Fields of the Extended_Active_EP_req Command**

Name	Type	Valid Range	Description
NWKAddrOfInterest	Device Address	16-bit NWK address	NWK address for the request.
startIndex	8 bits	0x00-0xff	Starting index within the Active Endpoint list in the response.

3516 **2.4.3.1.23.1 When Generated**

3517 The Extended_Active_EP_req command is generated from a local device wishing to acquire the list of
3518 endpoints on a remote device with simple descriptors. This command shall be unicast either to the remote
3519 device itself or to an alternative device that contains the discovery information of the remote device. The
3520 Extended_Active_EP_req is used for devices which support more active endpoints than can be returned by
3521 a single Active_EP_req.

3522 The local device shall generate the Extended_Active_EP_req command using the format illustrated in Ta-
3523 ble 2.68. The NWKAddrOfInterest field shall contain the network address of the remote device for which
3524 the active endpoint list is required. The startIndex field shall be set in the request to enable retrieval of lists
3525 of active endpoints from devices whose list exceeds the size of a single ASDU and where fragmentation is
3526 not supported.

3527 **2.4.3.1.23.2 Effect on Receipt**

3528 Upon receipt of this command, the recipient device shall process the command and generate an Extend-
3529 ed_Active_EP_rsp command in response, according to the description in section 2.4.4.2.21.1.

3530 The results in the Extended_Active_EP_rsp shall include the elements described in Table 2.68 with a se-
3531 lectable set of the list of active endpoints on the remote device starting with the entry StartIndex and con-
3532 tinuing with whole entries until the maximum APS packet length is reached or the application input and
3533 output cluster lists is exhausted, along with a status of SUCCESS.

2.4.3.2 End Device Bind, Bind, Unbind, and Bind Management Client Services Primitives

Table 2.69 lists the primitives supported by Device Profile: End Device Bind, Bind and Unbind Client Services. Each of these commands will be discussed in the following sections.

Table 2.69 End Device Bind, Bind, Unbind, and Bind Management Client Service Commands

End Device Bind, Bind and Unbind Client Services	Client Transmission	Server Processing
End_Device_Bind_req	O	O
Bind_req	O	O
Unbind_req	O	O
Bind_Register_req	O	O
Replace_Device_req	O	O
Store_Bkup_Bind_Entry_req	O	O
Remove_Bkup_Bind_Entry_req	O	O
Backup_Bind_Table_req	O	O
Recover_Bind_Table_req	O	O
Backup_Source_Bind_req	O	O
Recover_Source_Bind_req	O	O

2.4.3.2.1 End_Device_Bind_req

The End_Device_Bind_req command (ClusterID=0x0020) shall be formatted as illustrated in Figure 2.43.

Figure 2.43 Format of the End_Device_Bind_req Command Frame

Octets: 2	8	1	2	1	Variable	1	Variable
Binding Target	SrcIEEE Address	Src Endpoint	Profile ID	Num InClusters	InCluster List	Num OutClusters	OutCluster List

Table 2.70 specifies the fields of the End_Device_Bind_req command frame.

Table 2.70 Fields of the End_Device_Bind_req Command

Name	Type	Valid Range	Description
BindingTarget	Device Address	16-bit NWK Address	The address of the target for the binding. This can be either the primary binding cache device or the short address of the local device.
SrcIEEEAddress	IEEE Address	A valid 64-bit IEEE Address	The IEEE address of the device generating the request.
SrcEndpoint	8 bits	1-254	The endpoint on the device generating the request.
ProfileID	Integer	0x0000-0xffff	ProfileID which is to be matched between two End_Device_Bind_req received at the ZigBee Coordinator within the timeout value pre-configured in the ZigBee Coordinator.
NumInClusters	Integer	0x00-0xff	The number of Input Clusters provided for end device binding within the InClusterList.
InClusterList	2 bytes * NumInClusters		List of Input ClusterIDs to be used for matching. The InClusterList is the desired list to be matched by the ZigBee coordinator with the Remote Device's output clusters (the elements of the InClusterList are supported input clusters of the Local Device).
NumOutClusters	Integer	0x00-0xff	The number of Output Clusters provided for matching within OutClusterList.
OutClusterList	2 bytes * NumOutClusters		List of Output ClusterIDs to be used for matching. The OutClusterList is the desired list to be matched by the ZigBee coordinator with the Remote Device's input clusters (the elements of the OutClusterList are supported output clusters of the Local Device).

2.4.3.2.1.1 When Generated

The End_Device_Bind_req is generated from a Local Device wishing to perform End Device Bind with a Remote Device. The End_Device_Bind_req is generated, typically based on some user action like a button press. The destination addressing on this command shall be unicast, and the destination address shall be that of the ZigBee Coordinator.

2.4.3.2.1.2 Effect on Receipt

On receipt of this command, the ZigBee coordinator shall first check that the supplied endpoint is within the specified range. If the supplied endpoint does not fall within the specified range, the ZigBee coordinator shall return an End_Device_Bind_rsp with a status of INVALID_EP.

3554 If the supplied endpoint is within the specified range, the ZigBee Coordinator shall retain the
 3555 End_Device_Bind_req for a pre-configured timeout duration awaiting a second End_Device_Bind_req. If
 3556 the second request does not appear within the timeout period, the ZigBee Coordinator shall generate a
 3557 TIMEOUT status and return it with the End_Device_Bind_rsp to the originating Local Device. Assuming
 3558 the second End_Device_Bind_req is received within the timeout period, it shall be matched with the first
 3559 request on the basis of the ProfileID, InClusterList and OutClusterList.

3560 If no match of the ProfileID is detected by using the Profile Id Endpoint Matching Rules (see section
 3561 2.3.3.2), or if the ProfileID matches but none of the InClusterList or OutClusterList elements match, a
 3562 status of NO_MATCH shall be supplied to both Local Devices via End_Device_Bind_rsp to each device. If a
 3563 match of Profile ID and at least one input or output clusterID is detected, an End_Device_Bind_rsp with
 3564 status SUCCESS shall be issued to each Local Device which generated the End_Device_Bind_req.

3565 In order to facilitate a toggle action, the ZigBee Coordinator shall then issue an Unbind_req command to
 3566 the BindingTarget, specifying any one of the matched ClusterID values. If the returned status value is
 3567 NO_ENTRY, the ZigBee Coordinator shall issue a Bind_req command for each matched ClusterID value.
 3568 Otherwise, the ZigBee Coordinator shall conclude that the binding records are instead to be removed and
 3569 shall issue an Unbind_req command for any further matched ClusterID values.

3570 The initial Unbind_req and any subsequent Bind_reqs or Unbind_reqs containing the matched clusters shall
 3571 be directed to one of the BindingTargets specified by the generating devices. The BindingTarget is selected
 3572 on an individual basis for each matched cluster, as the Binding Target selected by the generating device
 3573 having that cluster as an output cluster. The SrcAddress field shall contain the 64-bit IEEE address of that
 3574 same generating device and the SrcEndp field shall contain its endpoint. The DstAddress field shall contain
 3575 the 64-bit IEEE address of the generating device having the matched cluster in its input cluster list and the
 3576 DstEndp field shall contain its endpoint.

2.4.3.2.2 Bind_req

3577 The Bind_req command (ClusterID=0x0021) shall be formatted as illustrated in Figure 2.44.

3579 **Figure 2.44 Format of the Bind_req Command Frame**

Octets: 8	1	2	1	2/8	0/1
SrcAddress	SrcEndp	ClusterID	DstAddrMode	DstAddress	DstEndp

3580 Table 2.71 specifies the fields of the Bind_req command frame.

3581 **Table 2.71 Fields of the Bind_req Command**

Name	Type	Valid Range	Description
SrcAddress	IEEE Address	A valid 64-bit IEEE address	The IEEE address for the source.
SrcEndp	Integer	0x01-0xfe	The source endpoint for the binding entry.
ClusterID	Integer	0x0000-0xffff	The identifier of the cluster on the source device that is bound to the destination.

Name	Type	Valid Range	Description
DstAddrMode	Integer	0x00-0xff	The addressing mode for the destination address used in this command. This field can take one of the non-reserved values from the following list: 0x00 = reserved 0x01 = 16-bit group address for DstAddress and DstEndp not present 0x02 = reserved 0x03 = 64-bit extended address for DstAddress and DstEndp present 0x04 – 0xff = reserved
DstAddress	Address	As specified by the DstAddrMode field	The destination address for the binding entry.
DstEndp	Integer	0x01-0xfe	This field shall be present only if the DstAddrMode field has a value of 0x03 and, if present, shall be the destination endpoint for the binding entry.

3582 **2.4.3.2.2.1 When Generated**

3583 The Bind_req is generated from a Local Device wishing to create a Binding Table entry for the source and
 3584 destination addresses contained as parameters. The destination addressing on this command shall be unicast
 3585 only, and the destination address shall be that of a Primary binding table cache or to the SrcAddress itself.
 3586 The Binding Manager is optionally supported on the source device (unless that device is also the ZigBee
 3587 Coordinator) so that device shall issue a NOT_SUPPORTED status to the Bind_req if not supported.

3588 **2.4.3.2.2.2 Effect on Receipt**

3589 Upon receipt, a Remote Device (a Primary binding table cache or the device designated by SrcAddress)
 3590 shall create a Binding Table entry based on the parameters supplied in the Bind_req if the Binding Manager
 3591 is supported. If the remote device is a primary binding table cache, the following additional processing is
 3592 required. First, the primary cache shall check its table of devices holding their own source bindings for the
 3593 device in SrcAddress and, if it is found, shall issue another Bind_req to that device with the same entry.
 3594 Second, the primary cache shall check if there is a backup binding table cache and, if so, shall issue a
 3595 Store_Bkup_Binding_Entry_req command to backup the new entry. The Remote Device shall then respond
 3596 with SUCCESS if the entry has been created by the Binding Manager; otherwise, the Remote Device shall
 3597 respond with NOT_SUPPORTED.

3598

3599 **2.4.3.2.3 Unbind_req**

3600 The Unbind_req command (ClusterID=0x0022) shall be formatted as illustrated in Figure 2.45.

3601 **Figure 2.45 Format of the Unbind_req Command Frame**

Octets: 8	1	2	1	2/8	0/1
SrcAddress	SrcEndp	ClusterID	DstAddrMode	DstAddress	DstEndp

3602
 3603
 3604

Table 2.72 specifies the fields of the Unbind_req command frame.

Table 2.72 Fields of the Unbind_req Command

Name	Type	Valid Range	Description
SrcAddress	IEEE Address	A valid 64-bit IEEE address	The IEEE address for the source
SrcEndp	Integer	0x01-0xfe	The source endpoint for the binding entry
ClusterID	Integer	0x0000-0xffff	The identifier of the cluster on the source device that is bound to the destination.
DstAddrMode	Integer	0x00-0xff	The addressing mode for the destination address used in this command. This field can take one of the non-reserved values from the following list: 0x00 = reserved 0x01 = 16-bit group address for DstAddress and DstEndp not present 0x02 = reserved 0x03 = 64-bit extended address for DstAddress and DstEndp present 0x04 – 0xff = reserved
DstAddress	Address	As specified by the DstAddrMode field	The destination address for the binding entry.
DstEndp	Integer	0x01-00xfe	This field shall be present only if the DstAddrMode field has a value of 0x03 and, if present, shall be the destination endpoint for the binding entry.

3605 **2.4.3.2.3.1 When Generated**

3606 The Unbind_req is generated from a Local Device wishing to remove a Binding Table entry for the source
 3607 and destination addresses contained as parameters. The destination addressing on this command shall be
 3608 unicast only and the destination address must be that of the Primary binding table cache or the SrcAddress.

3609 **2.4.3.2.3.2 Effect on Receipt**

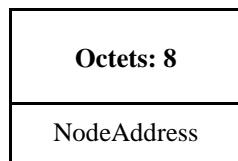
3610 The Remote Device shall evaluate whether this request is supported. If the request is not supported, a Status
 3611 of NOT_SUPPORTED shall be returned. If the request is supported, the Remote Device (a Primary binding
 3612 table cache or the SrcAddress) shall remove a Binding Table entry based on the parameters supplied in the
 3613 Unbind_req. If the Remote Device is a primary binding table cache, the following additional processing is
 3614 required. First, the primary cache shall check its table of devices holding their own source bindings for the
 3615 device in SrcAddress and, if it is found, shall issue another Unbind_req to that device with the same entry.
 3616 Second, the primary cache shall check if there is a backup binding table cache and, if so, shall issue a
 3617 Remove_Bkup_Bind_Entry_req command to remove the backup of this entry. If a Binding Table entry for
 3618 the SrcAddress, SrcEndp, ClusterID, DstAddress, DstEndp contained as parameters does not exist, the
 3619 Remote Device shall respond with NO_ENTRY. Otherwise, the Remote Device shall delete the indicated
 3620 Binding Table entry and respond with SUCCESS.

3621

3622 **2.4.3.2.4 Bind_Register_req**

3623 The Bind_Register_req command (ClusterID=0x0023) shall be formatted as illustrated in Figure 2.46.

3624 **Figure 2.46 Format of the Bind_Register_req Command Frame**



3625

3626 Table 2.73 specifies the fields for the Bind_Register_req command frame.

3627 **Table 2.73 Fields of the Bind_Register_req Command**

Name	Type	Valid Range	Description
NodeAddress	IEEE Address	A valid 64-bit IEEE address	The address of the node wishing to hold its own binding table.

3628 **2.4.3.2.4.1 When Generated**

3629 The Bind_Register_req is generated from a Local Device and sent to a primary binding table cache device
3630 to register that the local device wishes to hold its own binding table entries. The destination addressing
3631 mode for this request is unicast.

3632 **2.4.3.2.4.2 Effect on Receipt**

3633 If the remote device is not a primary binding table cache it shall return a status of NOT_SUPPORTED.
3634 Otherwise, the primary binding table cache shall add the NodeAddress given by the parameter to its table
3635 of source devices which have chosen to store their own binding table. If this fails, it shall return a status of
3636 TABLE_FULL. Otherwise, it returns a status of SUCCESS. If an entry for the NodeAddress already exists
3637 in the table of source devices, the behavior will be the same as if it had been newly added. The source de-
3638 vice should clear its source binding table before issuing this command to avoid synchronization problems.
3639 In the successful case, any existing bind entries from the binding table whose source address is NodeAd-
3640 dress will be sent to the requesting device for inclusion in its source binding table. See Bind_Register_rsp
3641 for further details. Subsequent bind entries written to the binding list will cause copies to be written to the
3642 source device using Bind_req.

3643 **2.4.3.2.5 Replace_Device_req**

3644 The Replace_Device_req command (ClusterID=0x0024) shall be formatted as illustrated in Figure 2.47.

3645 **Figure 2.47 Format of the Replace_Device_req Command Frame**

Octets: 8	1	8	1
OldAddress	OldEndpoint	NewAddress	NewEndpoint

3646

3647 Table 2.74 specifies the fields for the Replace_Device_req command frame.

3648

Table 2.74 Fields of the Replace_Device_req Command

Name	Type	Valid Range	Description
OldAddress	IEEE Address	A valid 64-bit IEEE	The address of the node being replaced.
OldEndpoint	Integer	0x00 - 0xfe	The endpoint being replaced.
NewAddress	IEEE Address	A valid 64-bit IEEE	The replacement address.
NewEndpoint	Integer	0x01 - 0xfe	The replacement endpoint.

3649

2.4.3.2.5.1 When Generated

3650 The Replace_Device_req is intended for use by a special device such as a Commissioning tool and is sent
 3651 to a primary binding table cache device to change all binding table entries which match OldAddress and
 3652 OldEndpoint as specified. Note that OldEndpoint = 0 has special meaning and signifies that only the ad-
 3653 dress needs to be matched. The endpoint in the binding table will not be changed in this case and so New-
 3654 Endpoint is ignored. The processing changes all binding table entries for which the source address is the
 3655 same as OldAddress and, if OldEndpoint is non-zero, for which the source endpoint is the same as
 3656 OldEndpoint. It shall also change all binding table entries which have the destination address the same as
 3657 OldAddress and, if OldEndpoint is non-zero, the destination endpoint the same as OldEndpoint. The desti-
 3658 nation addressing mode for this request is unicast.

3659

2.4.3.2.5.2 Effect on Receipt

3660 If the remote device is not a primary binding table cache, it shall return a status of NOT_SUPPORTED.
 3661 The primary binding table cache shall check if the OldAddress parameter is non-zero and, if so, shall search
 3662 its binding table for entries of source addresses and source endpoint, or destination addresses and destina-
 3663 tion endpoint, that are set the same as OldAddress and OldEndpoint. It shall change these entries to have
 3664 NewAddress and NewEndpoint. In the case that OldEndpoint is zero, the primary binding table cache shall
 3665 search its binding table for entries whose source address or destination address match OldAddress. It shall
 3666 change these entries to have NewAddress leaving the endpoint value unchanged and ignoring NewEnd-
 3667 point. It shall then return a response of SUCCESS. The primary binding table cache shall also be responsi-
 3668 ble for notifying affected devices which are registered as holding their own source binding table of the
 3669 changes. This will be necessary for each changed binding table entry, where the destination address was
 3670 changed and the source address appears in the list of source devices which have chosen to store their own
 3671 binding table. In each of these cases, the amended binding table entry will be sent to the source device us-
 3672 ing an Unbind_req command for the old entry followed by a Bind_req command for the new one. In the
 3673 case that the source address of the bind entry has been changed, it will be necessary for the primary binding
 3674 table cache to send an Unbind_req command to the old source device if it is a source bind device and to
 3675 send a Bind_req command to the new source bind device if it is a source bind device. The primary binding
 3676 table cache shall also update the backup binding table cache by means of the Re-
 3677 move_bkup_binding_entry_req command for the old entry and Store_bkup_binding_entry_req for the al-
 3678 tered entry.

3679

2.4.3.2.6 Store_Bkup_Bind_Entry_req

3680
 3681

The Store_Bkup_Bind_Entry_req command (ClusterID=0x0025) shall be formatted as illustrated in Figure
 2.48.

3682

Figure 2.48 Format of the Store_Bkup_Bind_Entry_req Command Frame

Octets: 8	1	2	1	2/8	0/1
SrcAddress	SrcEndp	ClusterID	DstAddrMode	DstAddress	DstEndp

3683

3684 Table 2.75 specifies the fields of the Store_Bkup_Bind_Entry_req command frame.

Table 2.75 Fields of the Store_Bkup_Bind_Entry_req Command

Name	Type	Valid Range	Description
SrcAddress	IEEE Address	A valid 64-bit IEEE address	The IEEE address for the source.
SrcEndpoint	Integer	0x01 - 0xfe	The source endpoint for the binding entry.
ClusterId	Integer	0x0000 - 0xffff	The identifier of the cluster on the source device that is bound to the destination.
DstAddrMode	Integer	0x00-0xff	The addressing mode for the destination address used in this command. This field can take one of the non-reserved values from the following list: 0x00 = reserved 0x01 = 16-bit group address for DstAddress and DstEndp not present 0x02 = reserved 0x03 = 64-bit extended address for DstAddress and DstEndp present 0x04 – 0xff = reserved
DstAddress	Address	As specified by the DstAddrMode field	The destination address for the binding entry.
DstEndp	Integer	0x01-0xfe	This field shall be present only if the DstAddrMode field has a value of 0x03 and, if present, shall be the destination endpoint for the binding entry.

3686

2.4.3.2.6.1 When Generated

3687
3688
3689
3690

The Store_Bkup_Bind_Entry_req is generated from a local primary binding table cache and sent to a remote backup binding table cache device to request backup storage of the entry. It will be generated whenever a new binding table entry has been created by the primary binding table cache. The destination addressing mode for this request is unicast.

3691 **2.4.3.2.6.2 Effect on Receipt**

3692 If the remote device is not a backup binding table cache it shall return a status of NOT_SUPPORTED. If it
3693 is the backup binding table cache, it should maintain the identity of the primary binding table cache from
3694 previous discovery. If the contents of the Store_Bkup_Bind_Entry parameters match an existing entry in
3695 the binding table cache, then the remote device shall return SUCCESS. Otherwise, the backup binding table
3696 cache shall add the binding entry to its binding table and return a status of SUCCESS. If there is no room, it
3697 shall return a status of TABLE_FULL.

3698 **2.4.3.2.7 Remove_Bkup_Bind_Entry_req**

3699 The Remove_Bkup_Bind_Entry_req command (ClusterID=0x0026) shall be formatted as illustrated in
3700 Figure 2.49.

3701 **Figure 2.49 Format of the Remove_Bkup_Bind_Entry_req Command Frame**

Octets: 8	1	2	1	2/8	0/1
SrcAddress	SrcEndp	ClusterID	DstAddrMode	DstAddress	DstEndp

3702

3703 Table 2.76 specifies the fields of the Remove_Bkup_Bind_Entry_req command frame.

3704 **Table 2.76 Fields of the Remove_Bkup_Bind_Entry_req Command**

Name	Type	Valid Range	Description
SrcAddress	IEEE Address	A valid 64-bit IEEE address	The IEEE address for the source.
SrcEndpoint	Integer	0x01 - 0xfe	The endpoint for the binding entry.
ClusterId	Integer	0x0000 - 0xffff	The identifier of the cluster on the source device that is bound to the destination.
DstAddrMode	Integer	0x00-0xff	The addressing mode for the destination address used in this command. This field can take one of the non-reserved values from the following list: 0x00 = reserved 0x01 = 16-bit group address for DstAddress and DstEndp not present 0x02 = reserved 0x03 = 64-bit extended address for DstAddress and DstEndp present 0x04 – 0xff = reserved
DstAddress	Address	As specified by the DstAddrMode field	The destination address for the binding entry.

Name	Type	Valid Range	Description
DstEndp	Integer	0x01-0xfe	This field shall be present only if the DstAddrMode field has a value of 0x03 and, if present, shall be the destination endpoint for the binding entry.

3705 **2.4.3.2.7.1 When Generated**

3706 The Remove_Bkup_Bind_Entry_req is generated from a local primary binding table cache and sent to a
 3707 remote backup binding table cache device to request removal of the entry from backup storage. It will be
 3708 generated whenever a binding table entry has been unbound by the primary binding table cache. The desti-
 3709 nation addressing mode for this request is unicast.

3710 **2.4.3.2.7.2 Effect on Receipt**

3711 If the remote device is not a backup binding table cache, it shall return a status of NOT_SUPPORTED. If it
 3712 is a backup binding table cache, it should maintain the identity of the primary binding table cache from
 3713 previous discovery. If it does not recognize the sending device as the primary binding table cache, it shall
 3714 return a status of INV_REQUESTTYPE. Otherwise, the backup binding table cache shall search its binding
 3715 table for the entry corresponding to the supplied parameters. If no entry is found, it shall return a status of
 3716 NO_ENTRY. Otherwise, it shall delete the entry and return a status of SUCCESS.

3717 **2.4.3.2.8 Backup_Bind_Table_req**

3718 The Backup_Bind_Table_req command (ClusterID=0x0027) shall be formatted as illustrated in Figure
 3719 2.50.

3720 **Figure 2.50 Format of the Backup_Bind_Table_req Command Frame**

Octets: 2	2	2	Variable
BindingTableEntries	StartIndex	BindingTableListCount	BindingTableList

3721

3722 Table 2.77 specifies the fields of the Backup_Bind_Table_req command frame.

3723 **Table 2.77 Fields of the Backup_Bind_Table_req Command**

Name	Type	Valid Range	Description
BindingTableEntries	Integer	0x0000 - 0xffff	Total number of binding table entries on the primary binding table cache device.
StartIndex	Integer	0x0000 - 0xffff	Starting index within the binding table of entries.
BindingTableListCount	Integer	0x0000 - 0xffff	Number of binding table entries included within BindingTableList.

Name	Type	Valid Range	Description
BindingTableList	List of binding descriptors	The list shall contain the number of elements given by the BindingTableListCount of the elements in the primary binding table cache devices's binding table (see Table 2.134 for details.)	A list of descriptors beginning with the StartIndex element and continuing for BindingTableListCount of the elements in the primary binding table cache devices's binding table (see Table 2.134 for details.)

3724 **2.4.3.2.8.1 When Generated**

3725 The Backup_Bind_Table_req is generated from a local primary binding table cache and sent to the remote
3726 backup binding table cache device to request backup storage of its entire binding table. The destination ad-
3727 dressing mode for this request is unicast.

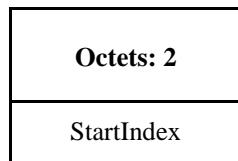
3728 **2.4.3.2.8.2 Effect on Receipt**

3729 If the remote device is not a backup binding table cache, it shall return a status of NOT_SUPPORTED. If it
3730 is a backup binding table cache, it should maintain the identity of the primary binding table cache from
3731 previous discovery. If it does not recognize the sending device as a primary binding table cache, it shall re-
3732 turn a status of INV_REQUESTTYPE. Otherwise, the backup binding table cache shall overwrite the
3733 binding entries in its binding table starting with StartIndex and continuing for BindingTableListCount en-
3734 tries. If this exceeds its table size, it shall fill in as many entries as possible and return a status of TA-
3735 BLE_FULL. Otherwise, it shall return a status of SUCCESS. The table is effectively truncated to the end of
3736 the last entry written by this request. The new size of the table is returned in the response and will be equal
3737 to
3738 StartIndex + BindingTableListCount unless TABLE_FULL is being returned it which case it will be the
3739 maximum size of the table.

3740 **2.4.3.2.9 Recover_Bind_Table_req**

3741 The Recover_Bind_Table_req command (ClusterID=0x0028) shall be formatted as illustrated in Figure
3742 2.51.

3743 **Figure 2.51 Fields of the Recover_Bind_Table_req Command Frame**



3744

3745 Table 2.78 specifies the fields of the Recover_Bind_Table_req command frame.

3746 **Table 2.78 Fields of the Recover_Bind_Table_req Command**

Name	Type	Valid Range	Description
StartIndex	Integer	0x0000 - 0xffff	Starting index for the requested elements of the binding table

3747 **2.4.3.2.9.1 When Generated**

3748 The Recover_Bind_Table_req is generated from a local primary binding table cache and sent to a remote
3749 backup binding table cache device when it wants a complete restore of the binding table. The destination
3750 addressing mode for this request is unicast.

3751 **2.4.3.2.9.2 Effect on Receipt**

3752 If the remote device is not the backup binding table cache, it shall return a status of NOT_SUPPORTED. If
3753 it does not recognize the sending device as a primary binding table cache it shall return a status of
3754 INV_REQUESTTYPE. Otherwise, the backup binding table cache shall prepare a list of binding table en-
3755 tries from its backup beginning with StartIndex. It will fit in as many entries as possible into a Recov-
3756 er_Bind_Table_rsp command and return a status of SUCCESS.

3757 **2.4.3.2.10 Backup_Source_Bind_req**

3758 The Backup_Source_Bind_req command (ClusterID=0x0029) shall be formatted as illustrated in Figure
3759 2.52.

3760 **Figure 2.52 Fields of the Backup_Source_Bind_req Command Frame**

Octets: 2	2	2	Variable
SourceTableEntries	StartIndex	SourceTableListCount	SourceTableList

3761

3762 Table 2.79 specifies the fields of the Backup_Source_Bind_req command frame.

3763 **Table 2.79 Fields of the Backup_Source_Bind_req Command**

Name	Type	Valid Range	Description
SourceTableEntries	Integer	0x0000 - 0xffff	Total number of source table entries on the primary binding table cache device.
startIndex	Integer	0x0000 - 0xffff	Starting index within the binding table of the entries in SourceTableList.
SourceTableListCount	Integer	0x0000 - 0xffff	Number of source table entries included within SourceTableList.
SourceTableList	List of IEEE Addresses	The list shall contain the number of elements given by the SourceTableListCount	A list of addresses beginning with the StartIndex element and continuing for SourceTableListCount of source addresses in the primary binding table cache device's source table.

3764 **2.4.3.2.10.1 When Generated**

3765 The Backup_Source_Bind_req is generated from a local primary binding table cache and sent to a remote
3766 backup binding table cache device to request backup storage of its entire source table. The destination ad-
3767 dressing mode for this request is unicast.

3768 **2.4.3.2.10.2 Effect on Receipt**

3769 If the remote device is not the backup binding table cache, it shall return a status of NOT_SUPPORTED. If
3770 it does not recognize the sending device as a primary binding table cache, it shall return a status of
3771 INV_REQUESTTYPE. Otherwise, the backup binding table cache shall overwrite the source entries in its
3772 backup source table starting with StartIndex and continuing for SourceTableListCount entries. If this ex-
3773 ceeds its table size, it shall return a status of TABLE_FULL. Otherwise, it shall return a status of SUCC-
3774 ESS. The command always truncates the backup table to a number of entries equal to its maximum size or
3775 SourceTableEntries, whichever is smaller.

3776 **2.4.3.2.11 Recover_Source_Bind_req**

3777 The Recover_Source_Bind_req command (ClusterID=0x002a) shall be formatted as illustrated in Figure
3778 2.53.

3779 **Figure 2.53 Format of the Recover_Source_Bind_req Command Frame**



3780

3781 Table 2.80 specifies the fields of the Recover_Source_Bind_req command frame.

3782 **Table 2.80 Fields of the Recover_Source_Bind_req Command**

Name	Type	Valid Range	Description
StartIndex	Integer	0x0000 - 0xffff	Starting index for the requested elements of the binding table

3783 **2.4.3.2.11.1 When Generated**

3784 The Recover_Source_Bind_req is generated from a local primary binding table cache and sent to the re-
3785 mote backup binding table cache device when it wants a complete restore of the source binding table. The
3786 destination addressing mode for this request is unicast.

3787 **2.4.3.2.11.2 Effect on Receipt**

3788 If the remote device is not the backup binding table cache it shall return a status of NOT_SUPPORTED. If
3789 it does not recognize the sending device as a primary binding table cache, it shall return a status of
3790 INV_REQUESTTYPE. Otherwise, the backup binding table cache shall prepare a list of source binding ta-
3791 ble entries from its backup beginning with StartIndex. It will fit in as many entries as possible into a Re-
3792 cover_Source_Bind_rsp command and return a status of SUCCESS.

3793 **2.4.3.3 Network Management Client Services**

3794 Table 2.81 lists the commands supported by Device Profile: Network Management Client Services. Each of
3795 these primitives will be discussed in the following sections.

3796

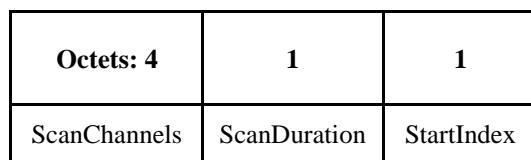
Table 2.81 Network Management Client Services Commands

Network Management Client Services	Client Transmission	Server Processing
Mgmt_NWK_Disc_req	O	O
Mgmt_Lqi_req	O	M
Mgmt_Rtg_req	O	O
Mgmt_Bind_req	O	M
Mgmt_Leave_req	O	M
Mgmt_Direct_Join_req	O	O
Mgmt_Permit_Joining_req	O	M
Mgmt_Cache_req	O	O
Mgmt_NWK_Update_req	O	O

3797 **2.4.3.3.1 Mgmt_NWK_Disc_req**

3798 The Mgmt_NWK_Disc_req command (ClusterID=0x0030) shall be formatted as illustrated in Figure 2.54.

3799 **Figure 2.54 Format of the Mgmt_NWK_Disc_req Command Frame**



3800

3801 Table 2.82 specifies the fields for the Mgmt_NWK_Disc_req command frame.

3802 **Table 2.82 Fields of the Mgmt_NWK_Disc_req Command**

Name	Type	Valid Range	Description
ScanChannels	Bitmap	32-bit field	See section 3.2.2.1 for details on NLME-NETWORK-DISCOVERY.request ScanChannels parameter.
ScanDuration	Integer	0x00-0x0e	A value used to calculate the length of time to spend scanning each channel. The time spent scanning each channel is (aBaseSuperframeDuration * (2 ⁿ + 1)) symbols, where n is the value of the ScanDuration parameter. For more information on MAC sub-layer scanning (see [B1]).

Name	Type	Valid Range	Description
StartIndex	Integer	0x00-0xff	Starting index within the resulting NLME-NETWORK-DISCOVERY.confirm NetworkList to begin reporting for the Mgmt_NWK_Disc_rsp.

3803 **2.4.3.3.1.1 When Generated**

3804 The Mgmt_NWK_Disc_req is generated from a Local Device requesting that the Remote Device execute a
3805 Scan to report back networks in the vicinity of the Local Device. The destination addressing on this com-
3806 mand shall be unicast.

3807 **2.4.3.3.1.2 Effect on Receipt**

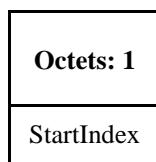
3808 The Remote Device shall execute an NLME-NETWORK-DISCOVERY.request using the ScanChannels
3809 and ScanDuration parameters supplied with the Mgmt_NWK_Disc_req command. The results of the Scan
3810 shall be reported back to the Local Device via the Mgmt_NWK_Disc_rsp command.

3811 If this command is not supported in the Remote Device, the return status provided with the
3812 Mgmt_NWK_Disc_rsp shall be NOT_SUPPORTED. If the scan was successful, the
3813 Mgmt_NWK_Disc_rsp command shall contain a status of SUCCESS and the results of the scan shall be
3814 reported, beginning with the StartIndex element of the NetworkList. If the scan was unsuccessful, the
3815 Mgmt_NWK_Disc_rsp command shall contain the error code reported in the
3816 NLME-NETWORK-DISCOVERY.confirm primitive.

3817 **2.4.3.3.2 Mgmt_Lqi_req**

3818 The Mgmt_Lqi_req command (ClusterID=0x0031) shall be formatted as illustrated in Figure 2.55.

3819 **Figure 2.55 Format of the Mgmt_Lqi_req Command Frame**



3820

3821 Table 2.83 specifies the fields for the Mgmt_NWK_Disc_req command frame.

3822 **Table 2.83 Fields of the Mgmt_Lqi_req Command**

Name	Type	Valid Range	Description
StartIndex	Integer	0x00-0xff	Starting Index for the requested elements of the Neighbor Table.

3823 **2.4.3.3.2.1 When Generated**

3824 The Mgmt_Lqi_req is generated from a Local Device wishing to obtain a neighbor list for the Remote De-
3825 vice along with associated LQI values to each neighbor. The destination addressing on this command shall
3826 be unicast only. It may be sent to a coordinator, router, or end device.

3827 **2.4.3.3.2.2 Effect on Receipt**

3828 Upon receipt, a Remote Device (ZigBee Router or ZigBee Coordinator) shall retrieve the entries of the
3829 neighbor table and associated LQI values via the NLME-GET.request primitive (for the *nwkNeighborTable*
3830 attribute) and report the resulting neighbor table (obtained via the NLME-GET.confirm primitive) via the
3831 Mgmt_Lqi_rsp command.

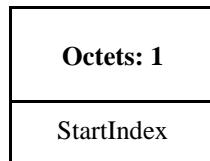
3832 Prior to revision 21 of this specification, server processing of this command was optional. Additionally
3833 end devices were not required to support the command. As a result some devices may return
3834 NOT_SUPPORTED.

3835 If this command is not supported in the Remote Device, the return status provided with the Mgmt_Lqi_rsp
3836 shall be NOT_SUPPORTED. If the neighbor table was obtained successfully, the Mgmt_Lqi_rsp command
3837 shall contain a status of SUCCESS and the neighbor table shall be reported, beginning with the element in
3838 the list enumerated as StartIndex. If the neighbor table was not obtained successfully, the Mgmt_Lqi_rsp
3839 command shall contain the error code reported in the NLME-GET.confirm primitive.

3840 **2.4.3.3.3 Mgmt_Rtg_req**

3841 The Mgmt_Rtg_req command (ClusterID=0x0032) shall be formatted as illustrated in Figure 2.56.

3842 **Figure 2.56 Format of the Mgmt_Rtg_req Command Frame**



3843

3844 Table 2.84 specifies the fields for the Mgmt_Rtg_req command frame.

3845 **Table 2.84 Fields of the Mgmt_Rtg_req Command**

Name	Type	Valid Range	Description
StartIndex	Integer	0x00-0xff	Starting Index for the requested elements of the Routing Table.

3846 **2.4.3.3.3.1 When Generated**

3847 The Mgmt_Rtg_req is generated from a Local Device wishing to retrieve the contents of the Routing Table
3848 from the Remote Device. The destination addressing on this command shall be unicast only and the desti-
3849 nation address must be that of the ZigBee Router or ZigBee Coordinator.

3850 **2.4.3.3.3.2 Effect on Receipt**

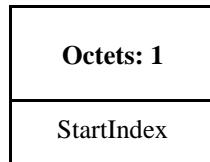
3851 Upon receipt, a Remote Device (ZigBee Coordinator or ZigBee Router) shall retrieve the entries of the
3852 routing table from the NWK layer via the NLME-GET.request primitive (for the *nwkRouteTable* attribute)
3853 and report the resulting routing table (obtained via the NLME-GET.confirm primitive) via the
3854 Mgmt_Rtg_rsp command.

3855 If the Remote Device does not support this optional management request, it shall return a Status of
3856 NOT_SUPPORTED. If the routing table was obtained successfully, the Mgmt_Rtg_req command shall
3857 contain a status of SUCCESS and the routing table shall be reported, beginning with the element in the list
3858 enumerated as StartIndex. If the routing table was not obtained successfully, the Mgmt_Rtg_rsp command
3859 shall contain the error code reported in the NLME-GET.confirm primitive.

3860 **2.4.3.3.4 Mgmt_Bind_req**

3861 The Mgmt_Bind_req command (ClusterID=0x0033) shall be formatted as illustrated in Figure 2.57.

3862 **Figure 2.57 Format of the Mgmt_Bind_req Command Frame**



3863

3864 Table 2.85 specifies the fields for the Mgmt_Bind_req command frame.

3865 **Table 2.85 Fields of the Mgmt_Bind_req Command**

Name	Type	Valid Range	Description
StartIndex	Integer	0x00-0xff	Starting Index for the requested elements of the Binding Table.

3866 **2.4.3.3.4.1 When Generated**

3867 The Mgmt_Bind_req is generated from a Local Device wishing to retrieve the contents of the Binding Ta-
3868 ble from the Remote Device. The destination addressing on this command shall be unicast only and the
3869 destination address must be that of a Primary binding table cache or source device holding its own binding
3870 table.

3871 **2.4.3.3.4.2 Effect on Receipt**

3872 Upon receipt, a Remote Device shall retrieve the entries of the binding table from the APS sub-layer via the
3873 APSME-GET.request primitive (for the *apsBindingTable* attribute) and report the resulting binding table
3874 (obtained via the APSME-GET.confirm primitive) via the Mgmt_Bind_rsp command.

3875 If the Remote Device does not support this optional management request, it shall return a status of
3876 NOT_SUPPORTED. If the binding table was obtained successfully, the Mgmt_Bind_rsp command shall
3877 contain a status of SUCCESS and the binding table shall be reported, beginning with the element in the list
3878 enumerated as StartIndex. If the binding table was not obtained successfully, the Mgmt_Bind_rsp com-
3879 mand shall contain the error code reported in the APSME-GET.confirm primitive.

3880 **2.4.3.3.5 Mgmt_Leave_req**

3881 The Mgmt_Leave_req command (ClusterID=0x0034) shall be formatted as illustrated in Figure 2.58.

3882 **Figure 2.58 Format of the Mgmt_Leave_req Command Frame**

Bits: 64	6	1	1
Device Address	Reserved	Remove Children	Rejoin

3883

3884 Table 2.86 specifies the fields for the Mgmt_Leave_req command frame.

3885

Table 2.86 Fields of the Mgmt_Leave_req Command

Name	Type	Valid Range	Description
DeviceAddress	Device Address	An extended 64-bit, IEEE address	See section 3.2.2.16 for details on the Device Address parameter within NLME-LEAVE.request. For DeviceAddress of NULL, a value of 0x0000000000000000 shall be used.
Remove Children	Bit	0 or 1	This field has a value of 1 if the device being asked to leave the network is also being asked to remove its child devices, if any. Otherwise, it has a value of 0.
Rejoin	Bit	0 or 1	This field has a value of 1 if the device being asked to leave from the current parent is requested to rejoin the network. Otherwise, it has a value of 0.

3886

2.4.3.3.5.1 When Generated

3887 The Mgmt_Leave_req is generated from a Local Device requesting that a Remote Device leave the network
3888 or to request that another device leave the network. The Mgmt_Leave_req is generated by a management
3889 application which directs the request to a Remote Device where the NLME-LEAVE.request is to be ex-
3890 ecuted using the parameter supplied by Mgmt_Leave_req.

3891

2.4.3.3.5.2 Effect on Receipt

3892 Upon receipt, the remote device shall process the leave request by executing the procedure in section
3893 3.6.1.10.3.1. If the leave request was validated and accepted, then the receiving device shall generate the
3894 NLME-LEAVE.request to disassociate from the currently associated network. The
3895 NLME-LEAVE.request shall have the DeviceAddress parameter set to the local device's *nwkIeeeAddress*
3896 from the NIB, the RemoveChildren shall be set to FALSE, and the Rejoin parameter shall be set to FALSE.

3897 The results of the leave attempt shall be reported back to the local device via the Mgmt_Leave_rsp com-
3898 mand.

3899 Versions of this specification prior to revision 21 did not mandate the requirement to support this com-
3900 mand. Therefore if the remote device did not support this optional management request, it would return a
3901 status of NOT_SUPPORTED. All devices certified against version 21 and later are now required to sup-
3902 port this command.

3903 If the leave attempt was executed successfully, the Mgmt_Leave_rsp command shall contain a status of
3904 SUCCESS. If the leave attempt was not executed successfully, the Mgmt_Leave_rsp command shall con-
3905 tain the error code reported in the NLME-LEAVE.confirm primitive.

3906

3907

2.4.3.3.6 Mgmt_Direct_Join_req

3908 The Mgmt_Direct_Join_req command (ClusterID=0x0035) shall be formatted as illustrated in Figure 2.59.

3910

Figure 2.59 Format of the Mgmt_Direct_Join_req Command Frame

Octets: 8	1
Device Address	Capability Information

3911

3912 Table 2.87 specifies the fields for the Mgmt_Direct_Join_req command frame.

3913

Table 2.87 Fields of the Mgmt_Direct_Join_req Command

Name	Type	Valid Range	Description
DeviceAddress	Device Address	An extended 64-bit, IEEE address	See section 3.2.2.14 for details on the DeviceAddress parameter within NLME-DIRECT-JOIN.request.
CapabilityInformation	Bitmap	See Table 3-47	The operating capabilities of the device being directly joined.

3914

2.4.3.3.6.1 When Generated

3915
3916
3917
3918

The Mgmt_Direct_Join_req is generated from a Local Device requesting that a Remote Device permit a device designated by DeviceAddress to join the network directly. The Mgmt_Direct_Join_req is generated by a management application which directs the request to a Remote Device where the NLME-DIRECT-JOIN.request is to be executed using the parameter supplied by Mgmt_Direct_Join_req.

3919

2.4.3.3.6.2 Effect on Receipt

3920
3921
3922
3923

Upon receipt, the remote device shall issue the NLME-DIRECT-JOIN.request primitive using the DeviceAddress and CapabilityInformation parameters supplied with the Mgmt_Direct_Join_req command. The results of the direct join attempt shall be reported back to the local device via the Mgmt_Direct_Join_rsp command.

3924
3925
3926
3927
3928

If the remote device does not support this optional management request, it shall return a status of NOT_SUPPORTED. If the direct join attempt was executed successfully, the Mgmt_Direct_Join_rsp command shall contain a status of SUCCESS. If the direct join attempt was not executed successfully, the Mgmt_Direct_Join_rsp command shall contain the error code reported in the NLME-DIRECT-JOIN.confirm primitive.

3929

2.4.3.3.7 Mgmt_Permit_Joining_req

3930
3931
3932

The Mgmt_Permit_Joining_req command (ClusterID=0x0036) shall be formatted as illustrated in Figure 2.60.

3933

Figure 2.60 Format of the Mgmt_Permit_Joining_req Command Frame

Octets: 1	1
PermitDuration	TC_Significance

3934

3935 Table 2.88 specifies the fields of the Mgmt_Permit_Joining_req command frame.

3936 **Table 2.88 Fields of the Mgmt_Permit_Joining_req Command**

Name	Type	Valid Range	Description
PermitDuration	Integer	0x00 - 0xfe	See section 3.2.2.5 for details on the PermitDuration parameter within NLME-PERMIT-JOINING.request.
TC_Significance	Boolean Integer	0x00 - 0x01	This field shall always have a value of 1, indicating a request to change the Trust Center policy. If a frame is received with a value of 0, it shall be treated as having a value of 1.

3937 **2.4.3.3.7.1 When Generated**

3938 The Mgmt_Permit_Joining_req is generated from a Local Device requesting that a remote device or devices allow or disallow association. The Mgmt_Permit_Joining_req is generated by a management application
3939 or commissioning tool which directs the request to a remote device(s) where the
3940 NLME-PERMIT-JOINING.request is executed using the PermitDuration parameter supplied by
3941 Mgmt_Permit_Joining_req. Additionally, if the remote device is the Trust Center and TC_Significance is
3942 set to 1, the Trust Center authentication policy will be affected. The addressing may be unicast or ‘broad-
3943 cast to all routers and coordinator.’

3945 **2.4.3.3.7.2 Effect on Receipt**

3946 Upon receipt, the remote device(s) shall issue the NLME-PERMIT-JOINING.request primitive using the
3947 PermitDuration parameter supplied with the Mgmt_Permit_Joining_req command. If the PermitDuration
3948 parameter is not equal to zero or 0xFF, the parameter is a number of seconds and joining is permitted until
3949 it counts down to zero, after which time, joining is not permitted. If the PermitDuration is set to zero, joining
3950 is not permitted.

3951 Versions of this specification prior to revision 21 allowed a value of 0xFF to be interpreted as ‘forever’.
3952 Version 21 and later do not allow this. All devices conforming to this specification shall interpret 0xFF as
3953 0xFE. Devices that wish to extend the PermitDuration beyond 0xFE seconds shall periodically re-send the
3954 Mgmt_Permit_Joining_req.

3955 If a second Mgmt_Permit_Joining_req is received while the previous one is still counting down, it will super-
3956 pcede the previous request.

3957 A value of zero for the TC_Significance field has been deprecated. The field shall always be included in
3958 the message and all received frames shall be treated as though set to 1, regardless of the actual received
3959 value. In other words, all Mgmt_Permit_Joining_req shall be treated as a request to change the TC Policy.
3960

3961 If the remote device is the Trust Center the Trust Center authorization policy may be affected. Whether the
3962 Trust Center accepts a change in its authorization policy is dependent upon its Trust Center policies. A
3963 Trust Center device receiving a Mgmt_Permit_Joining_req shall execute the procedure in section 4.7.3.2 to
3964 determine if the request is permitted. If the operation was not permitted, the status code of INVAILID_REQUEST
3965 shall be set. If the operation was allowed, the status code of SUCCESS shall be set.¹

3966 If the Mgmt_Permit_Joining_req primitive was received as a unicast, the results of the NLME-PERMIT-
3967 JOINING.request shall be reported back to the local device via the Mgmt_Permit_Joining_rsp command. If
3968 the command was received as a broadcast, no response shall be sent back.

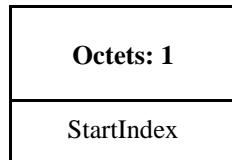
3969

¹ CCB 1550

3970 **2.4.3.3.8 Mgmt_Cache_req**

3971 The Mgmt_Cache_req command (ClusterID=0x0037) shall be formatted as illustrated in Figure 2.61.

3972 **Figure 2.61 Fields of the Mgmt_Cache_req Command Frame**



3973

3974 Table 2.89 specifies the fields of the Mgmt_Cache_req command frame.

3975

Table 2.89 Fields of the Mgmt_Cache_req Command

Name	Type	Valid Range	Description
startIndex	Integer	0x00 - 0xff	Starting Index for the requested elements of the discovery cache list.

3976 **2.4.3.3.8.1 When Generated**

3977 The Mgmt_Cache_req is provided to enable ZigBee devices on the network to retrieve a list of ZigBee End
3978 Devices registered with a Primary Discovery Cache device. The destination addressing on this primitive
3979 shall be unicast.

3980 **2.4.3.3.8.2 Effect on Receipt**

3981 Upon receipt, the Remote Device shall determine whether it is a Primary Discovery Cache or whether this
3982 optional request primitive is supported. If it is not a Primary Discovery Cache device or the
3983 Mgmt_Cache_req primitive is not supported, the Remote Device shall return a status of
3984 NOT_SUPPORTED. If the Remote Device is a Primary Discovery Cache and supports the
3985 Mgmt_Cache_req, the Remote Device shall return SUCCESS to the Local Device along with the discovery
3986 cache list which consists of the NWKAddr and IEEEAddr for each ZigBee End Device registered.

3987 **2.4.3.3.9 Mgmt_NWK_Update_req**

3988 The Mgmt_NWK_Update_req command (ClusterID=0x0038) shall be formatted as illustrated in Figure
3989 2.62.

3990

Figure 2.62 Fields of the Mgmt_NWK_Update_req Command Frame

Octets: 4	1	0/1	0/1	0/2
ScanChannels	ScanDuration	ScanCount	nwkUpdateId	nwkManagerAddr

3991

3992 Table 2.90 specifies the fields of the Mgmt_NWK_Update_req command frame.

3993

Table 2.90 Fields of the Mgmt_NWK_Update_req Command

Name	Type	Valid Range	Description
ScanChannels	Bitmap	32-bit field	See section 3.2.2.1 for details on NLME-ED-SCAN.request ScanChannels parameter.
ScanDuration	Integer	0x00-0x05 or 0xfe or 0xff	A value used to calculate the length of time to spend scanning each channel. The time spent scanning each channel is (aBaseSuperframeDuration * (2 ⁿ + 1)) symbols, where n is the value of the ScanDuration parameter. For more information on MAC sub-layer scanning (see [B1]). If ScanDuration has a value of 0xfe this is a request for channel change. If ScanDuration has a value of 0xff this is a request to change the <i>apsChannelMask</i> and <i>nwkManagerAddr</i> attributes.
ScanCount	Integer	0x00 - 0x05	This field represents the number of energy scans to be conducted and reported. This field shall be present only if the ScanDuration is within the range of 0x00 to 0x05.
nwkUpdateId	Integer	0x00 - 0xFF	The value of the <i>nwkUpdateId</i> contained in this request. This value is set by the Network Channel Manager prior to sending the message. This field shall only be present if the ScanDuration is 0xfe or 0xff. If the ScanDuration is 0xff, then the value in the <i>nwkUpdateID</i> shall be ignored.
nwkManagerAddr	Device Address	16-bit NWK address	This field shall be present only if the ScanDuration is set to 0xff, and, where present, indicates the NWK address for the device with the Network Manager bit set in its Node Descriptor.

3994

2.4.3.3.9.1 When Generated

3995
3996
3997

This command is provided to allow updating of network configuration parameters or to request information from devices on network conditions in the local operating environment. The destination addressing on this primitive shall be unicast or broadcast to all devices for which macRxOnWhenIdle = TRUE.

3998

2.4.3.3.9.2 Effect on Receipt

3999
4000
4001

Upon receipt, the Remote Device shall determine from the contents of the ScanDuration parameter whether this request is an update to the *apsChannelMask* and *nwkManagerAddr* attributes, a channel change command, or a request to scan channels and report the results.

4002
4003
4004
4005
4006
4007

If the ScanDuration parameter is equal to 0xfe, the message is a command to change channels. The receiver shall determine if the channel is one within the range of the current PHY, and if the command has a channel outside that range a response of INVALID_REQUEST shall be generated, and the original request shall be dropped and no further processing shall be done. This command provides a new active channel as a single channel in the ChannelMask in which case the APS IB is not updated. If the channel is valid for the current PHY then the procedure for channel change shall be initiated.

4008 If the ScanDuration parameter is equal to 0xff, the command provides a set of new apsChannelMask along
4009 with a new nwkManagerAddr. The Remote Device shall store the apsChannelMask in the APS IB and the
4010 nwkManagerAddr in the NIB without invocation of an NLME-ED-SCAN.request.
4011 If this command is unicast with ScanDuration set to 0xfe or 0xff, the Remote Device shall not respond. The
4012 network manager should request an APS acknowledgement in this case.
4013 If the ScanDuration is equal to 0x00 to 0x05 and the destination addressing on this command was unicast
4014 then the command is interpreted as a request to scan the channels described in ChannelMask. If the channel
4015 mask contains a channel outside the range of the current PHY, then a response of INVALID_REQUEST
4016 shall be sent back. Otherwise the device shall use the parameter ScanDuration and ScanCount, via invocation
4017 of an NLME-ED-SCAN.request. If the Remote Device does not support fragmentation and the resulting
4018 response will exceed the APDU, the Remote Device shall perform the Energy Detect Scan on as many of the
4019 requested channels as will fit into a single APDU, highlighting the list of actual scanned channels in the
4020 response parameter. If multiple scans are requested in the ScanCount, each scan is reported as a separate
4021 result. The Remote Device will employ an Energy Detect Scan using the request parameters, modified by the
4022 limitation described for fragmentation, and supply the results to the requesting device with a
4023 Mgmt_NWK_Update_notify with a SUCCESS status.
4024 Otherwise, if the ScanDuration is equal to 0x06 to 0xfd and the destination addressing on this command was
4025 unicast then the Remote Device shall return a status of INVALID_REQUEST.
4026 If the destination addressing on this command was not unicast then the Remote Device shall not transmit a
4027 response.

4028 **2.4.4 Server Services**

4029 The Device Profile Server Services support the processing of device and service discovery requests, end
4030 device bind requests, bind requests, unbind requests, and network management requests. Additionally,
4031 Server Services support transmission of these responses back to the requesting device.

4032 **2.4.4.1 ZDO Response Requirements**

4033 A device shall be required to support generation of the correct, corresponding ZDO response to all ZDO
4034 requests including ZDO messages defined in a future version of this specification. Server Processing
4035 marked optional in Table 2.91, Table 2.114, and Table 2.126 allow for the server to use NOT_SUPPORTED
4036 as the status code in the response to indicate the lack of support. ZDO requests unknown to the device shall
4037 be treated as unsupported and also use a NOT_SUPPORTED status code to indicate the device's lack of
4038 support for that feature. See below for construction of ZDO responses to unsupported requests. For all
4039 broadcast addressed requests (of any broadcast address type) to the server, if the command is not supported,
4040 the server shall drop the packet. No error status shall be unicast back to the Local Device for any broadcast
4041 addressed client request including, but not limited to, requests which are not supported on the server.

4042 For all unicast addressed requests to the server, if the command is not supported, the server shall formulate
4043 a response packet including the response Cluster ID and status fields only. The response Cluster ID shall be
4044 created by taking the request Cluster ID and setting the high order bit to create the response Cluster ID. The
4045 status field shall be set to NOT_SUPPORTED. The resulting response shall be unicast to the requesting
4046 client.

4047 2.4.4.2 Device and Service Discovery Server

4048 Table 2.91 lists the commands supported by the Device and Service Discovery Server Services device pro-
4049 file. Each of these commands will be discussed in the following sections. For receipt of the Device_annce
4050 command, the server shall check all internal references to the IEEE and 16-bit NWK addresses supplied in
4051 the request. For all references to the IEEE address in the Local Device, the corresponding NWK address
4052 supplied in the Device_annce shall be substituted. For any other references to the NWK address in the Lo-
4053 cal Device, the corresponding entry shall be marked as not having a known valid 16-bit NWK address,
4054 even if the IEEEAddr field in the message carries the value of 0xffffffffffff. The server shall not supply
4055 a response to the Device_annce.

4056 **Table 2.91 Device and Service Discovery Server Service Primitives**

Device and Service Discovery Server Services	Server Processing	Server Generation
NWK_addr_rsp	M	M
IEEE_addr_rsp	M	M
Node_Desc_rsp	M	M
Power_Desc_rsp	M	M
Simple_Desc_rsp	M	M
Active_EP_rsp	M	M
Match_Desc_rsp	M	M
Complex_Desc_rsp	O	M
User_Desc_rsp	O	M
User_Desc_conf	O	M
Parent_annce_rsp	M	M
System_Server_Discovery_rsp	O	M
Discovery_store_rsp	O	M
Node_Desc_store_rsp	O	M
Power_Desc_store_rsp	O	M
Active_EP_store_rsp	O	M
Simple_Desc_store_rsp	O	M

Device and Service Discovery Server Services	Server Processing	Server Generation
Remove_node_cache_rsp	O	M
Find_node_cache_rsp	O	M
Extended_Simple_Desc_rsp	O	M
Extended_Active_EP_rsp	O	M

4057 For Server Generation requirements see section 2.4.4.1.

4058

2.4.4.2.1 NWK_addr_rsp

4060 The NWK_addr_rsp command (ClusterID=0x8000) shall be formatted as illustrated in Figure 2.63.

4061

Figure 2.63 Format of the NWK_addr_rsp Command Frame

Octets: 1	8	2	0/1	0/1	Variable
Status	IEEEAddr RemoteDev	NWKAddr RemoteDev	Num AssocDev	StartIndex	NWKAddr AssocDevList

4062

4063 Table 2.92 specifies the fields of the NWK_addr_rsp command frame.

Table 2.92 Fields of the NWK_addr_rsp Command

Name	Type	Valid Range	Description
Status	Integer	SUCCESS, INV_REQUESTTYPE, or DEVICE_NOT_FOUND	The status of the NWK_addr_req command.
IEEEAddrRemoteDev	Device Ad- dress	An extended 64-bit, IEEE ad- dress	64-bit address for the Remote De- vice.
NWKAddrRemoteDev	Device Ad- dress	A 16-bit, NWK address	16-bit address for the Remote De- vice.

Name	Type	Valid Range	Description
NumAssocDev	Integer	0x00-0xff	<p>Count of the number of 16-bit short addresses to follow.</p> <p>If the RequestType in the request is Extended Response and there are no associated devices on the Remote Device, this field shall be set to 0.</p> <p>If an error occurs or the Request Type in the request is for a Single Device Response, this field shall not be included in the frame.</p>
startIndex	Integer	0x00-0xff	<p>Starting index into the list of associated devices for this report.</p> <p>If the RequestType in the request is Extended Response and there are no associated devices on the Remote Device, this field shall not be included in the frame.</p> <p>If an error occurs or the Request Type in the request is for a Single Device Response, this field shall not be included in the frame.</p>
NWKAddrAssocDevList	Device Address List	List of NumAssocDev 16-bit short addresses, each with range 0x0000 - 0xffff	<p>A list of 16-bit addresses, one corresponding to each associated device to Remote Device; The number of 16-bit network addresses contained in this field is specified in the NumAssocDev field.</p> <p>If the RequestType in the request is Extended Response and there are no associated devices on the Remote Device, this field shall not be included in the frame.</p> <p>If an error occurs or the Request Type in the request is for a Single Device Response, this field shall not be included in the frame.</p>

4065

2.4.4.2.1.1 When Generated

4066
4067
4068

The NWK_addr_rsp is generated by a Remote Device in response to a NWK_addr_req command inquiring as to the NWK address of the Remote Device or the NWK address of an address held in the neighbor table (see section 2.4.3.1.1.2 for a detailed description). The destination addressing on this command is unicast.

4069 **2.4.4.2.1.2 Effect on Receipt**

4070 On receipt of the NWK_addr_rsp command, the recipient is either notified of the status of its attempt to
4071 discover a NWK address from an IEEE address or notified of an error. If the NWK_addr_rsp command is
4072 received with a Status of SUCCESS, the remaining fields of the command contain the appropriate discov-
4073 ery information, according to the RequestType as specified in the original NWK_Addr_req command.
4074 Otherwise, the Status field indicates the error and the NumAssocDev, StartIndex, and NWKAddrAs-
4075 socDevList fields shall not be included.

4076 **2.4.4.2.2 IEEE_addr_rsp**

4077 The IEEE_addr_rsp command (ClusterID=0x8001) shall be formatted as illustrated in Figure 2.64.

4078 **Figure 2.64 Format of the IEEE_addr_rs Command Frame**

Octets: 1	8	2	0/1	0/1	Variable
Status	IEEEAddr RemoteDev	NWKAddr RemoteDev	NumAssocDev	StartIndex	NWKAddr AssocDevList

4079

4080 Table 2.93 specifies the fields of the IEEE_addr_rs command frame.

4081 **Table 2.93 IEEE_addr_rsp Parameters**

Name	Type	Valid Range	Description
Status	Integer	SUCCESS, INV_REQUESTTYPE or DEVICE_NOT_FOUND	The status of the IEEE_addr_req command.
IEEEAddrRemoteDev	Device Address	An extended 64-bit, IEEE address	64-bit address for the Remote Device.
NWKAddrRemoteDev	Device Address	A 16-bit, NWK address	16-bit address for the Remote Device.
NumAssocDev	Integer	0x00-0xff	Count of the number of 16-bit short ad- dresses to follow. If the RequestType in the request is Ex- tended Response and there are no associ- ated devices on the Remote Device, this field shall be set to 0. If an error occurs or the RequestType in the request is for a Single Device Re- sponse, this field shall not be included in the frame.

Name	Type	Valid Range	Description
StartIndex	Integer	0x00-0xff	<p>Starting index into the list of associated devices for this report.</p> <p>If the RequestType in the request is Extended Response and there are no associated devices on the Remote Device, this field shall not be included in the frame.</p> <p>If an error occurs or the RequestType in the request is for a Single Device Response, this field shall not be included in the frame.</p>
NWKAddrAssocDevList	Device Address List	<p>List of NumAssocDev 16-bit short addresses, each with range 0x0000 - 0xffff</p>	<p>A list of 16-bit addresses, one corresponding to each associated device to Remote Device; The number of 16-bit network addresses contained in this field is specified in the NumAssocDev field.</p> <p>If the RequestType in the request is Extended Response and there are no associated devices on the Remote Device, this field shall not be included in the frame.</p> <p>If an error occurs or the RequestType in the request is for a Single Device Response, this field shall not be included in the frame</p>

4082 **2.4.4.2.2.1 When Generated**

4083 The IEEE_addr_rsp is generated by a Remote Device in response to an IEEE_addr_req command inquiring
 4084 as to the 64-bit IEEE address of the Remote Device or the 64-bit IEEE address of an address held in a local
 4085 discovery cache (see section 2.4.3.1.2.2 for a detailed description). The destination addressing on this
 4086 command shall be unicast.

4087 **2.4.4.2.2.2 Effect on Receipt**

4088 On receipt of the IEEE_addr_rsp command, the recipient is either notified of the status of its attempt to
 4089 discover an IEEE address from an NWK address or notified of an error. If the IEEE_addr_rsp command is
 4090 received with a Status of SUCCESS, the remaining fields of the command contain the appropriate discov-
 4091 ery information, according to the RequestType as specified in the original IEEE_Addr_req command. Oth-
 4092 erwise, the Status field indicates the error and the NumAssocDev, StartIndex, and NWKAddrAssocDevList
 4093 fields shall not be included.

4094 **2.4.4.2.3 Node_Desc_rsp**

4095 The Node_Desc_rsp command (ClusterID=0x8002) shall be formatted as illustrated in Figure 2.65.

4096 **Figure 2.65 Format of the Node_Desc_rsp Command Frame**

Octets: 1	2	See section 2.3.2.3
Status	NWKAddrOfInterest	Node Descriptor

4098 Table 2.94 specifies the fields of the Node_Desc_rsp command frame.

4099 **Table 2.94 Fields of the Node_Desc_rsp Command**

Name	Type	Valid Range	Description
Status	Integer	SUCCESS, DEVICE_NOT_FOUND, INV_REQUESTTYPE, or NO_DESCRIPTOR	The status of the Node_Desc_req command.
NWKAddrOfInterest	Device Address	16-bit NWK address	NWK address for the request.
NodeDescriptor	Node Descriptor		See the Node Descriptor for- mat in section 2.3.2.3. This field shall only be included in the frame if the status field is equal to SUCCESS.

4100 **2.4.4.2.3.1 When Generated**

4101 The Node_Desc_rsp is generated by a remote device in response to a Node_Desc_req directed to the re-
4102 mote device. This command shall be unicast to the originator of the Node_Desc_req command.

4103 The remote device shall generate the Node_Desc_rsp command using the format illustrated in Table 2.94
4104 Fields of the Node_Desc_rsp Command. The NWKAddrOfInterest field shall match that specified in the
4105 original Node_Desc_req command. If the NWKAddrOfInterest field matches the network address of the
4106 remote device, it shall set the Status field to SUCCESS and include its node descriptor (see section 2.3.2.3)
4107 in the NodeDescriptor field.

4108 If the NWKAddrOfInterest field does not match the network address of the remote device and it is an end
4109 device, it shall set the Status field to INV_REQUESTTYPE and not include the NodeDescriptor field. If
4110 the NWKAddrOfInterest field does not match the network address of the remote device and it is the coor-
4111 dinator or a router, it shall determine whether the NWKAddrOfInterest field matches the network address
4112 of one of its children. If the NWKAddrOfInterest field does not match the network address of one of the
4113 children of the remote device, it shall set the Status field to DEVICE_NOT_FOUND and not include the
4114 NodeDescriptor field. If the NWKAddrOfInterest matches the network address of one of the children of the
4115 remote device, it shall determine whether a node descriptor for that device is available. If a node descriptor
4116 is not available for the child indicated by the NWKAddrOfInterest field, the remote device shall set the
4117 Status field to
4118 NO_DESCRIPTOR and not include the NodeDescriptor field. If a node descriptor is available for the child
4119 indicated by the NWKAddrOfInterest field, the remote device shall set the Status field to SUCCESS and
4120 include the node descriptor (see section 2.3.2.3) of the matching child device in the NodeDescriptor field.

4121 **2.4.4.2.3.2 Effect on Receipt**

4122 On receipt of the Node_Desc_rsp command, the recipient is either notified of the node descriptor of the
4123 remote device indicated in the original Node_Desc_req command or notified of an error. If the
4124 Node_Desc_rsp command is received with a Status of SUCCESS, the NodeDescriptor field shall contain
4125 the requested node descriptor. Otherwise, the Status field indicates the error and the NodeDescriptor field
4126 shall not be included.

4127 **2.4.4.2.4 Power_Desc_rsp**

4128 The Power_Desc_rsp command (ClusterID=0x8003) shall be formatted as illustrated in Figure 2.66.

4129

Figure 2.66 Format of the Power_Desc_rsp Command Frame

Octet: 1	2	Variable
Status	NWKAddrOfInterest	Power Descriptor

4130

4131 Table 2.95 specifies the fields of the Power_Desc_rsp command frame.

Table 2.95 Fields of the Power_Desc_rsp Command

Name	Type	Valid Range	Description
Status	Integer	SUCCESS, DEVICE_NOT_FOUND, INV_REQUESTTYPE or NO_DESCRIPTOR	The status of the Power_Desc_req command.
NWKAddrOfInterest	Device Address	16-bit NWK address	NWK address for the request.
PowerDescriptor	Power Descriptor		See the Node Power Descriptor format in section 2.3.2.4. This field shall only be included in the frame if the status field is equal to SUCCESS.

4133

2.4.4.2.4.1 When Generated

4134
4135

The Power_Desc_rsp is generated by a remote device in response to a Power_Desc_req directed to the remote device. This command shall be unicast to the originator of the Power_Desc_req command.

4136
4137
4138
4139

The remote device shall generate the Power_Desc_rsp command using the format illustrated in Table 2.95. The NWKAddrOfInterest field shall match that specified in the original Power_Desc_req command. If the NWKAddrOfInterest field matches the network address of the remote device, it shall set the Status field to SUCCESS and include its power descriptor (see section 2.3.2.4) in the PowerDescriptor field.

4140
4141
4142
4143
4144
4145
4146
4147
4148
4149
4150
4151
4152

If the NWKAddrOfInterest field does not match the network address of the remote device and it is an end device, it shall set the Status field to INV_REQUESTTYPE and not include the PowerDescriptor field. If the NWKAddrOfInterest field does not match the network address of the remote device and it is the coordinator or a router, it shall determine whether the NWKAddrOfInterest field matches the network address of one of its children. If the NWKAddrOfInterest field does not match the network address of one of the children of the remote device, it shall set the Status field to DEVICE_NOT_FOUND and not include the PowerDescriptor field. If the NWKAddrOfInterest matches the network address of one of the children of the remote device, it shall determine whether a power descriptor for that device is available. If a power descriptor is not available for the child indicated by the NWKAddrOfInterest field, the remote device shall set the Status field to NO_DESCRIPTOR and not include the PowerDescriptor field. If a power descriptor is available for the child indicated by the NWKAddrOfInterest field, the remote device shall set the Status field to SUCCESS and include the power descriptor (see section 2.3.2.4) of the matching child device in the PowerDescriptor field.

4153

2.4.4.2.4.2 Effect on Receipt

4154
4155
4156
4157
4158

On receipt of the Power_Desc_rsp command, the recipient is either notified of the power descriptor of the remote device indicated in the original Power_Desc_req command or notified of an error. If the Power_Desc_rsp command is received with a Status of SUCCESS, the PowerDescriptor field shall contain the requested power descriptor. Otherwise, the Status field indicates the error and the PowerDescriptor field shall not be included.

4159

2.4.4.2.5 Simple_Desc_rsp

4160

The Simple_Desc_rsp command (ClusterID=0x8004) shall be formatted as illustrated in Figure 2.67.

4161

Figure 2.67 Format of the Simple_Desc_rsp Command Frame

Octet: 1	2	1	Variable
Status	NWKAddrOfInterest	Length	Simple Descriptor

4162

4163 Table 2.96 specifies the fields of the Simple_Desc_rsp command frame.

4164

Table 2.96 Fields of the Simple_Desc_rsp Command

Name	Type	Valid Range	Description
Status	Integer	SUCCESS, INVALID_EP, NOT_ACTIVE, DEVICE_NOT_FOUND, INV_REQUESTTYPE or NO_DESCRIPTOR	The status of the Simple_Desc_req command.
NWKAddrOfInterest	Device Address	16-bit NWK address	NWK address for the request.
Length	Integer	0x00-0xff	Length in bytes of the Simple Descriptor to follow.
SimpleDescriptor	Simple Descriptor		See the Simple Descriptor format in section 2.3.2.5. This field shall only be included in the frame if the status field is equal to SUCCESS.

4165

2.4.4.2.5.1 When Generated

4166
4167

The Simple_Desc_rsp is generated by a remote device in response to a Simple_Desc_req directed to the remote device. This command shall be unicast to the originator of the Simple_Desc_req command.

4168
4169
4170
4171
4172

The remote device shall generate the Simple_Desc_rsp command using the format illustrated in Table 2.96. The NWKAddrOfInterest field shall match that specified in the original Simple_Desc_req command. If the endpoint field specified in the original Simple_Desc_req command does not fall within the correct range specified in Table 2.96 Fields of the Simple_Desc_req Command, the remote device shall set the Status field to INVALID_EP, set the Length field to 0 and not include the SimpleDescriptor field.

4173
4174
4175
4176
4177
4178
4179

If the NWKAddrOfInterest field matches the network address of the remote device, it shall determine whether the endpoint field specifies the identifier of an active endpoint on the device. If the endpoint field corresponds to an active endpoint, the remote device shall set the Status field to SUCCESS, set the Length field to the length of the simple descriptor on that endpoint, and include the simple descriptor (see section 2.3.2.5) for that endpoint in the SimpleDescriptor field. If the endpoint field does not correspond to an active endpoint, the remote device shall set the Status field to NOT_ACTIVE, set the Length field to 0, and not include the SimpleDescriptor field.

4180 If the NWKAddrOfInterest field does not match the network address of the remote device and it is an end
 4181 device, it shall set the Status field to INV_REQUESTTYPE, set the Length field to 0, and not include the
 4182 SimpleDescriptor field. If the NWKAddrOfInterest field does not match the network address of the remote
 4183 device and it is the coordinator or a router, it shall determine whether the NWKAddrOfInterest field
 4184 matches the network address of one of its children. If the NWKAddrOfInterest field does not match the
 4185 network address of one of the children of the remote device, it shall set the Status field to DE-
 4186 VICE_NOT_FOUND, set the Length field to 0, and not include the SimpleDescriptor field.

4187 If the NWKAddrOfInterest matches the network address of one of the children of the remote device, it shall
 4188 determine whether a simple descriptor for that device and on the requested endpoint is available. If a simple
 4189 descriptor is not available on the requested endpoint of the child indicated by the NWKAddrOfInterest
 4190 field, the remote device shall set the Status field to NO_DESCRIPTOR, set the Length field to 0, and not
 4191 include the SimpleDescriptor field. If a simple descriptor is available on the requested endpoint of the child
 4192 indicated by the NWKAddrOfInterest field, the remote device shall set the Status field to SUCCESS, set
 4193 the Length field to the length of the simple descriptor on that endpoint, and include the simple descriptor
 4194 (see section 2.3.2.5) for that endpoint of the matching child device in the SimpleDescriptor field.

4195 **2.4.4.2.5.2 Effect on Receipt**

4196 On receipt of the Simple_Desc_rsp command, the recipient is either notified of the simple descriptor on the
 4197 endpoint of the remote device indicated in the original Simple_Desc_req command or notified of an error.
 4198 If the Simple_Desc_rsp command is received with a Status of SUCCESS, the SimpleDescriptor field shall
 4199 contain the requested simple descriptor. Otherwise, the Status field indicates the error and the SimpleDe-
 4200 scriptor field shall not be included.

4201 **2.4.4.2.6 Active_EP_rsp**

4202 The Active_EP_rsp command (ClusterID=0x8005) shall be formatted as illustrated in Figure 2.68.

4203 **Figure 2.68 Format of the Active_EP_rsp Command Frame**

Octet: 1	2	1	Variable
Status	NWKAddrOfInterest	ActiveEPCount	ActiveEPList

4204 Table 2.97 specifies the fields of the Active_EP_rsp command frame.

4205 **Table 2.97 Fields of the Active_EP_rsp Command**

Name	Type	Valid Range	Description
Status	Integer	SUCCESS, DEVICE_NOT_FOUND, INV_REQUESTTYPE or NO_DESCRIPTOR	The status of the Active_EP_req com- mand.
NWKAddrOfInterest	Device Address	16-bit NWK address	NWK address for the request.
ActiveEPCount	Integer	0x00-0xff	The count of active endpoints on the Remote Device.
ActiveEPList			List of bytes each of which represents an 8-bit endpoint.

4206 **2.4.4.2.6.1 When Generated**

4207 The Active_EP_rsp is generated by a remote device in response to an Active_EP_req directed to the remote
4208 device. This command shall be unicast to the originator of the Active_EP_req command.

4209 The remote device shall generate the Active_EP_rsp command using the format illustrated in Table 2.97.
4210 The NWKAddrOfInterest field shall match that specified in the original Active_EP_req command. If the
4211 NWKAddrOfInterest field matches the network address of the remote device, it shall set the Status field to
4212 SUCCESS, set the ActiveEPCount field to the number of active endpoints on that device and include an
4213 ascending list of all the identifiers of the active endpoints on that device in the ActiveEPList field.

4214 If the NWKAddrOfInterest field does not match the network address of the remote device and it is an end
4215 device, it shall set the Status field to INV_REQUESTTYPE, set the ActiveEPCount field to 0, and not in-
4216 clude the ActiveEPList field. If the NWKAddrOfInterest field does not match the network address of the
4217 remote device and it is the coordinator or a router, it shall determine whether the NWKAddrOfInterest field
4218 matches the network address of a device it holds in a discovery cache. If the NWKAddrOfInterest field
4219 does not match the network address of a device it holds in a discovery cache, it shall set the Status field to
4220 DEVICE_NOT_FOUND, set the ActiveEPCount field to 0, and not include the ActiveEPList field. If the
4221 NWKAddrOfInterest matches the network address of a device held in a discovery cache on the remote de-
4222 vice, it shall determine whether that device has any active endpoints. If the discovery information corre-
4223 sponding to the ActiveEP request has not yet been uploaded to the discovery cache, the remote device shall
4224 set the Status field to NO_DESCRIPTOR, set the ActiveEPCount field to 0 and not include the ActiveEPList field.
4225 If the cached device has no active endpoints, the remote device shall set the Status field to
4226 SUCCESS, set the ActiveEPCount field to 0, and not include the ActiveEPList field. If the cached device
4227 has active endpoints, the remote device shall set the Status field to SUCCESS, set the ActiveEPCount field
4228 to the number of active endpoints on that device, and include an ascending list of all the identifiers of the
4229 active endpoints on that device in the ActiveEPList field.

4230 **2.4.4.2.6.2 Effect on Receipt**

4231 On receipt of the Active_EP_rsp command, the recipient is either notified of the active endpoints of the
4232 remote device indicated in the original Active_EP_req command or notified of an error. If the Active_EP_rsp
4233 command is received with a Status of SUCCESS, the ActiveEPCount field indicates the num-
4234 ber of entries in the ActiveEPList field. Otherwise, the Status field indicates the error and the ActiveEPList
4235 field shall not be included.

4236 **2.4.4.2.7 Match_Desc_rsp**

4237 The Match_Desc_rsp command (ClusterID=0x8006) shall be formatted as illustrated in Figure 2.69.

4238 **Figure 2.69 Format of the Match_Desc_rsp Command Frame**

Octet: 1	2	1	Variable
Status	NWKAddrOfInterest	Match Length	Match List

4239

4240 Table 2.98 specifies the fields of the Match_Desc_rsp command frame.

4241

Table 2.98 Fields of the Match_Desc_rsp Command

Name	Type	Valid Range	Description
Status	Integer	SUCCESS, DEVICE_NOT_FOUND, INV_REQUESTTYPE or NO_DESCRIPTOR	The status of the Match_Desc_req command.
NWKAddrOfInterest	Device Address	16-bit NWK address	NWK address for the request.
MatchLength	Integer	0x00-0xff	The count of endpoints on the Remote Device that match the request criteria.
MatchList			List of bytes each of which represents an 8-bit endpoint.

- 4242 **2.4.4.2.7.1 When Generated**
- 4243 The Match_Desc_rsp is generated by a remote device in response to a Match_Desc_req either broadcast or unicast to the remote device. This command shall be unicast to the originator of the Match_Desc_req command.
- 4244
- 4245
- 4246 The following describes the procedure for processing the Match_Desc_req and generation of Match_Desc_rsp.
- 4247
- 4248
- 4249 1. Set MatchLength to 0 and create an empty list MatchList.
- 4250 2. If the receiving device is an End Device and the NWKAddrOfInterest within the Match_Desc_req message does not match the nwkNetworkAddress of the NIB and is not a broadcast address, the following shall be performed. Otherwise it shall proceed to step 3.
- 4251
- 4252 a. If the NWK destination of the message is a broadcast address, no further processing shall be done.
- 4253
- 4254 b. If the NWK destination is a unicast address, the following shall be performed.
- 4255
- 4256 i. Set the Status value to INV_REQUESTTYPE.
- 4257 ii. Set the MatchLength to 0.
- 4258 iii. Construct a Match_Desc_Rsp with only Status and MatchLength fields.
- 4259 iv. Send the message as a unicast to the source of the Match_Desc_req.
- 4260 v. No further processing shall be done.
- 4261 3. If the NWKAddrOfInterest is equal to the nwkNetworkAddress of the NIB, or is a broadcast address, perform the following procedure. Otherwise proceed to step 4.
- 4262
- 4263 a. Apply the match criteria in section 2.4.4.2.7.1 for all local Simple Descriptors.
- 4264 b. For each Simple Descriptor that matches with at least one cluster, add the endpoint once to MatchList and increment MatchLength.
- 4265

- 4266 4. If the NWKAddrOfInterest is not a broadcast address, the NWKAddressOfInterest is not equal to
4267 the nwkNetworkAddress of the local NIB, and the device is a coordinator or router, then the fol-
4268 lowing shall be performed. Otherwise proceed to step 5.
- 4269 a. Examine each entry in the nwkNeighborTable and perform the following procedure.
- 4270 i. If the Network Address of the entry does not match the NWKAddrOfInterest or
4271 the Device Type is not equal to 0x02 (ZigBee End Device), do not process this
4272 entry. Continue to the next entry in the nwkNeighborTable.
- 4273 ii. If no cached Simple Descriptors for the device are available, skip this device and
4274 proceed to the next entry in the nwkNeighborTable.
- 4275 iii. Apply the match criteria in section 2.4.4.2.7.1.1 for each cached Simple De-
4276 scriptor.
- 4277 iv. For each endpoint that matches with at least once cluster, add that endpoint once
4278 to the MatchList and increment MatchLength.
- 4279 v. Proceed to step 7.
- 4280 b. If the NWKAddrOfInterest does not match any entry in the nwkNeighborTable, perform
4281 the following:
- 4282 i. Set the Status to DEVICE_NOT_FOUND.
- 4283 ii. Construct a Match_Desc_Rsp with Status and MatchLength fields only.
- 4284 iii. Unicast the message to the source of the Match_Desc_req.
- 4285 iv. No further processing shall take place.
- 4286 5. If the MatchLength is 0 and the NWK destination of the Match_Desc_Req was a broadcast ad-
4287 dress, no further processing shall be done. Otherwise proceed to step 6.
- 4288 6. If the MatchLength is 0 and the NWKAddrOfInterest matched an entry in the nwkNeighborTable,
4289 the following shall be performed. Otherwise proceed to step 7.
- 4290 a. Set the Status to NO_DESCRIPTOR
- 4291 b. Construct a Match_Desc_Rsp with Status and MatchLength only.
- 4292 c. Unicast the Match_Desc_Rsp to the source of the Match_Desc_Req.
- 4293 d. No further processing shall be done.
- 4294 7. The following shall be performed. This is the case for both MatchLength > 0 and MatchLength
4295 == 0.
- 4296 a. Set the Status to SUCCESS.
- 4297 b. Construct a Match_Desc_Rsp with Status, NWKAddrOfInterest, MatchLength, and
4298 MatchList.
- 4299 c. Unicast the response to the NWK source of the Match_Desc_Req.

4301 **2.4.4.2.7.1.1 Simple Descriptor Matching Rules**

4302 These rules will examine a ProfileID, InputClusterList, OutputClusterList, and a SimpleDescriptor. The
4303 following shall be performed:

- 4304 1. The device shall first check if the ProfileID field matches using the Profile ID of the SimpleDe-
4305 scriptor and the Endpoint Matching Rules (see section 2.3.3.2). If the profile identifiers do not
4306 match, the device shall note the match as unsuccessful and perform no further processing.

- 4307 2. Examine the InputClusterList and compare each item to the Application Input Cluster List of the
4308 SimpleDescriptor.
4309 a. If a cluster ID matches exactly, then the device shall note the match as successful and
4310 perform no further matching. Processing is complete.
4311 3. Examine the OutputClusterList and compare each item to the Application Output Cluster List of the
4312 SimpleDescriptor.
4313 a. If a cluster ID matches exactly, then the device shall note the match as successful and
4314 perform no further matching. Processing is complete.
4315 4. The device shall note the match as unsuccessful. Processing is complete.

4316

4317 **2.4.4.2.7.2 Effect on Receipt**

4318 On receipt of the Match_Desc_rsp command, the recipient is either notified of the results of its match criterion
4319 query indicated in the original Match_Desc_req command or notified of an error. If the
4320 Match_Desc_rsp command is received with a Status of SUCCESS, the MatchList field shall contain the list
4321 of endpoints containing simple descriptors that matched the criterion. Otherwise, the Status field indicates
4322 the error and the MatchList field shall not be included.

4323 **2.4.4.2.8 Complex_Desc_rsp**

4324 The Complex_Desc_rsp command (ClusterID=0x8010) shall be formatted as illustrated in Figure 2.70.

4325 **Figure 2.70 Format of the Complex_Desc_rsp Command Frame**

Octet: 1	2	1	Variable
Status	NWKAddrOfInterest	Length	Complex Descriptor

4326

4327 Table 2.99 specifies the fields of the Complex_Desc_rsp command frame.

4328 **Table 2.99 Fields of the Complex_Desc_rsp Command**

Name	Type	Valid Range	Description
Status	Integer	SUCCESS, NOT_SUPPORTED, DEVICE_NOT_FOUND, INV_REQUESTTYPE or NO_DESCRIPTOR	The status of the Complex_Desc_req command.
NWKAddrOfInterest	Device Ad- dress	16-bit NWK address	NWK address for the request.
Length	Integer	0x00-0xff	Length in bytes of the ComplexDescriptor field.

Name	Type	Valid Range	Description
ComplexDescriptor	Complex Descriptor		See the Complex Descriptor format in section 2.3.2.6. This field shall only be included in the frame if the status field is equal to SUCCESS.

4329 **2.4.4.2.8.1 When Generated**

4330 The Complex_Desc_rsp is generated by a remote device in response to a Complex_Desc_req directed to
4331 the remote device. This command shall be unicast to the originator of the Complex_Desc_req command.

4332 The remote device shall generate the Complex_Desc_rsp command using the format illustrated in Table
4333 2.99. The NWKAddrOfInterest field shall match that specified in the original Complex_Desc_req com-
4334 mand. If the NWKAddrOfInterest field matches the network address of the remote device but a complex
4335 descriptor does not exist, it shall set the Status field to NOT_SUPPORTED, set the Length field to 0, and
4336 not include the ComplexDescriptor field. If the NWKAddrOfInterest field matches the network address of
4337 the remote device and a complex descriptor exists, it shall set the Status field to SUCCESS, set the Length
4338 field to the length of the complex descriptor, and include its complex descriptor (see section
4339 2.3.2.6Complex Descriptor) in the ComplexDescriptor field.

4340 If the NWKAddrOfInterest field does not match the network address of the remote device and it is an end
4341 device, it shall set the Status field to INV_REQUESTTYPE, set the Length field to 0, and not include the
4342 ComplexDescriptor field. If the NWKAddrOfInterest field does not match the network address of the re-
4343 mote device and it is the coordinator or a router, it shall determine whether the NWKAddrOfInterest field
4344 matches the network address of one of its children. If the NWKAddrOfInterest field does not match the
4345 network address of one of the children of the remote device, it shall set the Status field to DE-
4346 VICE_NOT_FOUND, set the Length field to 0, and not include the ComplexDescriptor field. If the
4347 NWKAddrOfInterest matches the network address of one of the children of the remote device, it shall de-
4348 termine whether a complex descriptor for that device is available. If a complex descriptor is not available
4349 for the child indicated by the NWKAddrOfInterest field, the remote device shall set the Status field to
4350 NO_DESCRIPTOR, set the Length field to 0, and not include the ComplexDescriptor field. If a complex
4351 descriptor is available for the child indicated by the NWKAddrOfInterest field, the remote device shall set
4352 the Status field to SUCCESS, set the Length field to the length of the complex descriptor for that device,
4353 and include the complex descriptor (see section 2.3.2.6) of the matching child device in the Com-
4354 plexDescriptor field.

4355 **2.4.4.2.8.2 Effect on Receipt**

4356 On receipt of the Complex_Desc_rsp command, the recipient is either notified of the complex descriptor of
4357 the remote device indicated in the original Complex_Desc_req command or notified of an error. If the
4358 Complex_Desc_rsp command is received with a Status of SUCCESS, the ComplexDescriptor field shall
4359 contain the requested complex descriptor. Otherwise, the Status field indicates the error and the Com-
4360 plexDescriptor field shall not be included.

4361 **2.4.4.2.9 User_Desc_rsp**

4362 The User_Desc_rsp command (ClusterID=0x8011) shall be formatted as illustrated in Figure 2.71.

4363

Figure 2.71 Format of the User_Desc_rsp Command Frame

Octet: 1	2	1	Variable
Status	NWKAddrOfInterest	Length	User Descriptor

4364

Table 2.100 specifies the fields of the User_Desc_rsp command frame.

4365

Table 2.100 Fields of the User_Desc_rsp Command

Name	Type	Valid Range	Description
Status	Integer	SUCCESS, NOT_SUPPORTED, DEVICE_NOT_FOUND, INV_REQUESTTYPE or NO_DESCRIPTOR	The status of the Us- er_Desc_req command.
NWKAddrOfInterest	Device Address	16-bit NWK address	NWK address for the re- quest.
Length	Integer	0x00-0x10	Length in bytes of the UserDescriptor field.
UserDescriptor	User Descriptor		See the User Descriptor format in section 2.3.2.7. This field shall only be in- cluded in the frame if the status field is equal to SUCCESS.

4366

2.4.4.2.9.1 When Generated

4367
4368

The User_Desc_rsp is generated by a remote device in response to a User_Desc_req directed to the remote device. This command shall be unicast to the originator of the User_Desc_req command.

4369
4370
4371
4372
4373
4374
4375

The remote device shall generate the User_Desc_rsp command using the format illustrated in Table 2.100. The NWKAddrOfInterest field shall match that specified in the original User_Desc_req command. If the NWKAddrOfInterest field matches the network address of the remote device but a user descriptor does not exist, it shall set the Status field to NO_DESCRIPTOR, set the Length field to 0, and not include the UserDescriptor field. If the NWKAddrOfInterest field matches the network address of the remote device and a user descriptor exists, it shall set the Status field to SUCCESS, set the Length field to the length of the user descriptor, and include its user descriptor (see section 2.3.2.7) in the UserDescriptor field.

4376 If the NWKAddrOfInterest field does not match the network address of the remote device and it is an end
4377 device, it shall set the Status field to INV_REQUESTTYPE, set the Length field to 0, and not include the
4378 UserDescriptor field. If the NWKAddrOfInterest field does not match the network address of the remote
4379 device and it is the coordinator or a router, it shall determine whether the NWKAddrOfInterest field
4380 matches the network address of one of its children. If the NWKAddrOfInterest field does not match the
4381 network address of one of the children of the remote device, it shall set the Status field to DE-
4382 VICE_NOT_FOUND, set the Length field to 0, and not include the UserDescriptor field. If the NWKAd-
4383 drOfInterest matches the network address of one of the children of the remote device, it shall determine
4384 whether a user descriptor for that device is available. If a user descriptor is not available for the child indi-
4385 cated by the NWKAddrOfInterest field, the remote device shall set the Status field to NO_DESCRIPTOR,
4386 set the Length field to 0, and not include the UserDescriptor field. If a user descriptor is available for the
4387 child indicated by the
4388 NWKAddrOfInterest field, the remote device shall set the Status field to SUCCESS, set the Length field to
4389 the length of the user descriptor for that device, and include the user descriptor (see section 2.3.2.7) of the
4390 matching child device in the UserDescriptor field.

4391 **2.4.4.2.9.2 Effect on Receipt**

4392 On receipt of the User_Desc_rsp command, the recipient is either notified of the user descriptor of the re-
4393 mote device indicated in the original User_Desc_req command or notified of an error. If the User_Desc_rsp
4394 command is received with a Status of SUCCESS, the UserDescriptor field shall contain the requested user
4395 descriptor. Otherwise, the Status field indicates the error and the UserDescriptor field shall not be included.

4396 **2.4.4.2.10 System_Server_Discovery_rsp**

4397 The System_Server_Discovery_rsp command (ClusterID=0x8015) shall be formatted as illustrated in Fig-
4398 ure 2.72.

4399 **Figure 2.72 System_Server_Discovery_rsp Command Frame**

Octet: 1	2
Status	ServerMask

4400

4401 Table 2.101 specifies the fields of the System_Server_Discovery_rsp command frame.

4402

Table 2.101 Fields of the System_Server_Discovery_rsp Command

Name	Type	Valid Range	Description
Status	Integer	SUCCESS	The status of the System_Server_Discovery_rsp command.
ServerMask	Integer	Bitmap	See Table 2.32 for bit assignments.

4403

2.4.4.2.10.1 When Generated

4404 The System_Server_Discovery_rsp is generated from Remote Devices on receipt of a System_Server_Discovery_req primitive if the parameter matches the Server Mask field in its node descriptor. If there is no
 4405 match, the System_Server_Discovery_req shall be ignored and no response given. Matching is performed
 4406 by masking the ServerMask parameter of the System_Server_Discovery_req with the Server Mask field in
 4407 the node descriptor. This command shall be unicast to the device which sent System_Server_Discovery_req
 4408 with Acknowledge request set in TxOptions. The parameter ServerMask contains the bits in the parameter
 4409 of the request which match the server mask in the node descriptor.
 4410

4411

2.4.4.2.10.2 Effect on Receipt

4412 The requesting device is notified that this device has some of the system server functionality that the re-
 4413 questing device is seeking.

4414 If the Network Manager bit was set in the System_Server_Discovery_rsp, then the Remote Device's NWK
 4415 address shall be set into the *nwkManagerAddr* of the NIB.

4416

2.4.4.2.11 User_Desc_conf

4417 The User_Desc_conf command (ClusterID=0x8014) shall be formatted as illustrated in Figure 2.73.

4418

Figure 2.73 Format of the User_Desc_conf Command Frame

Octets: 1	2
Status	NWKAddrOfInterest

4419

Table 2.102 specifies the fields of the User_Desc_conf command frame.

4420

Table 2.102 Fields of the User_Desc_conf Command

Name	Type	Valid Range	Description
Status	Integer	SUCCESS, NOT_SUPPORTED, DEVICE_NOT_FOUND, INV_REQUESTTYPE or NO_DESCRIPTOR	The status of the User_Desc_set command.
NWKAddrOfInterest	Device Address	Any 16-bit NWK address	The network address of the device on which the user descriptor set attempt was made.

4421

2.4.4.2.11.1 When Generated

4422 The User_Desc_conf is generated by a remote device in response to a User_Desc_set directed to the remote
4423 device. This command shall be unicast to the originator of the User_Desc_set command.

4424 The remote device shall generate the User_Desc_conf command using the format illustrated in Table 2.102.
4425 The NWKAddrOfInterest field shall match that specified in the original User_Desc_set command. If the
4426 NWKAddrOfInterest field matches the network address of the remote device but a user descriptor does not
4427 exist, it shall set the Status field to NOT_SUPPORTED. If the NWKAddrOfInterest field matches the net-
4428 work address of the remote device and a user descriptor exists, it shall set the Status field to SUCCESS and
4429 configure the user descriptor with the ASCII character string specified in the original User_Desc_set com-
4430 mand.

4431 If the NWKAddrOfInterest field does not match the network address of the remote device and it is an end
4432 device, it shall set the Status field to INV_REQUESTTYPE. If the NWKAddrOfInterest field does not
4433 match the network address of the remote device and it is the coordinator or a router, it shall determine
4434 whether the NWKAddrOfInterest field matches the network address of one of its children. If the NWKAd-
4435 drOfInterest field does not match the network address of one of the children of the remote device, it shall
4436 set the Status field to DEVICE_NOT_FOUND. If the NWKAddrOfInterest matches the network address of
4437 one of the children of the remote device, it shall determine whether a user descriptor for that device is
4438 available. If a user descriptor is not available for the child indicated by the NWKAddrOfInterest field, the
4439 remote device shall set the Status field to NO_DESCRIPTOR. If a user descriptor is available for the child
4440 indicated by the NWKAddrOfInterest field, the remote device shall set the Status field to SUCCESS and
4441 configure the user descriptor with the ASCII character string specified in the original User_Desc_set com-
4442 mand.

4443

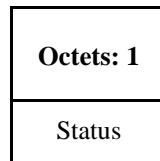
2.4.4.2.11.2 Effect on Receipt

4444 The local device is notified of the results of its attempt to configure the user descriptor on a remote device.

2.4.4.2.12 Discovery_Cache_rsp

4445 The Discovery_Cache_rsp command (ClusterID=0x8012) shall be formatted as illustrated in Figure 2.74.

4446 **Figure 2.74 Format of the Discovery_Cache_rsp Command Frame**



4447

4449 Table 2.103 specifies the fields of the Discovery_Cache_rsp Command Frame.

4450 **Table 2.103 Fields of the Discovery_Cache_rsp Command**

Name	Type	Valid Range	Description
Status	Integer	SUCCESS	The status of the Discovery_Cache_req command.

4451 **2.4.4.2.12.1 When Generated**

4452 The Discovery_Cache_rsp is generated by Primary Discovery Cache devices receiving the
4453 Discovery_Cache_req. Remote Devices which are not Primary Discovery Cache devices (as designated in
4454 its Node Descriptor) should not respond to the Discovery_Cache_req command.

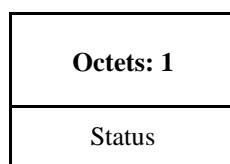
4455 **2.4.4.2.12.2 Effect on Receipt**

4456 Upon receipt of the Discovery_Cache_rsp, the Local Device determines if a SUCCESS status was returned.
4457 If no Discovery_Cache_rsp messages were returned from the original Discovery_Cache_req command,
4458 then the Local Device should increase the radius for the request to locate Primary Discovery Cache devices
4459 beyond the radius supplied in the previous request. If a SUCCESS status is returned, the Local Device
4460 should use the Discovery_Store_req, targeted to the Remote Device supplying the response, to determine
4461 whether sufficient discovery cache storage is available.

4462 **2.4.4.2.13 Discovery_store_rsp**

4463 The Discovery_store_rsp command (ClusterID=0x8016) shall be formatted as illustrated in Figure 2.75.

4464 **Figure 2.75 Format of the Discovery_store_rsp Command Frame**



4465

4466 Table 2.104 specifies the fields of the Discovery_store_rsp command frame.

4467 **Table 2.104 Fields of the Discovery_store_rsp Command**

Name	Type	Valid Range	Description
Status	Integer	SUCCESS, INSUFFICIENT_SPACE or NOT_SUPPORTED	The status of the Discovery_store_req command.

4468 **2.4.4.2.13.1 When Generated**

4469 The Discovery_store_rsp is provided to notify a Local Device of the request status from a Primary Discovery Cache device. Included in the response is a status code to notify the Local Device whether the request is
4470 successful (the Primary Cache Device has space to store the discovery cache data for the Local Device),
4471 whether the request is unsupported (meaning the Remote Device is not a Primary Discovery Cache device),
4472 or insufficient space exists.
4473

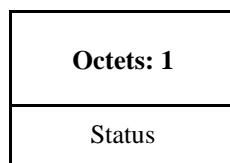
4474 **2.4.4.2.13.2 Effect on Receipt**

4475 Upon receipt, the Local Device shall determine whether the response status indicates that the Remote De-
4476 vice is not a Primary Cache Device as indicated by a NOT_SUPPORTED status. If a NOT_SUPPORTED
4477 status is returned, the Local Device should process any other Discovery_store_rsp devices from other Re-
4478 mote Devices or re-perform the Discovery_Cache_req to determine the address of another Primary Discov-
4479 ery Cache device (eliminating the address of the Remote Device that responded with NOT_SUPPORTED
4480 if it responds again to the Discovery_Cache_req). If an INSUFFICIENT_SPACE status is returned, the
4481 Local Device should also process any other Discovery_store_rsp and re-perform the Discovery_Cache_req
4482 if none of the responses indicate SUCCESS (with the radius field increased to include more Remote De-
4483 vices). If a
4484 SUCCESS status is returned, the Local Device shall upload its discovery cache information to the Remote
4485 Device via the Node_Desc_store_req, Power_Desc_store_req, Active_EP_store_req, and
4486 Simple_Desc_store_req.

4487 **2.4.4.2.14 Node_Desc_store_rsp**

4488 The Node_Desc_store_rsp command (ClusterID=0x8017) shall be formatted as illustrated in Figure 2.76.

4489 **Figure 2.76 Format of the Node_Desc_store_rsp Command Frame**



4490

4491 Table 2.105 specifies the fields of the Node_Desc_store_rsp command frame.

4492 **Table 2.105 Fields of the Node_Desc_store_rsp Command**

Name	Type	Valid Range	Description
Status	Integer	SUCCESS, INSUFFICIENT_SPACE, NOT_PERMITTED or NOT_SUPPORTED	The status of the Node_store_rsp command.

4493 **2.4.4.2.14.1 When Generated**

4494 The Node_store_rsp is provided to notify a Local Device of the request status from a Primary Discovery
4495 Cache device. Included in the response is a status code to notify the Local Device whether the request is
4496 successful (the Primary Cache Device has space to store the discovery cache data for the Local Device),
4497 whether the request is not supported (meaning the Remote Device is not a Primary Discovery Cache de-
4498 vice), or insufficient space exists.

4499 **2.4.4.2.14.2 Effect on Receipt**

4500 Upon receipt, the Local Device shall determine whether the response status indicates that the Remote De-
4501 vice is not a Primary Cache Device as indicated by a NOT_SUPPORTED status. If a NOT_SUPPORTED
4502 status is returned, the Local Device should re-perform discovery of the Primary Discovery Cache device. If
4503 a NOT_PERMITTED status is returned, the local device must first issue a Discovery_store_req with a re-
4504 turned SUCCESS status. If an INSUFFICIENT_SPACE status is returned, the Local Device shall also send
4505 the Remote Device a Remove_node_cache_req. If a SUCCESS status is returned, the Local Device should
4506 continue to upload its remaining discovery cache information to the Remote Device via the Pow-
4507 er_Desc_store_req, Active_EP_store_req, and Simple_Desc_store_req.

4508 **2.4.4.2.15 Power_Desc_store_rsp**

4509 The Power_Desc_store_rsp command (ClusterID=0x8018) shall be formatted as illustrated in Figure 2.77.

4510 **Figure 2.77 Format of the Power_Desc_store_rsp Command Frame**

Octets: 1	8	Variable
Status	IEEEAddr	PowerDescriptor

4511
4512 Table 2.106 specifies the fields of the Power_Desc_store_rsp command frame.

4513 **Table 2.106 Fields of the Power_Desc_store_rsp Command**

Name	Type	Valid Range	Description
Status	Integer	SUCCESS INSUFFICIENT_SPACE, NOT_PERMITTED or NOT_SUPPORTED	The status of the Power_store_rsp command.

4514 **2.4.4.2.15.1 When Generated**

4515 The Power_Desc_store_rsp is provided to notify a Local Device of the request status from a Primary Dis-
4516covery Cache device. Included in the response is a status code to notify the Local Device whether the re-
4517quest is successful (the Primary Cache Device has space to store the discovery cache data for the Local De-
4518vice), whether the request is not supported (meaning the Remote Device is not a Primary Discovery Cache
4519device), or insufficient space exists.

4520 **2.4.4.2.15.2 Effect on Receipt**

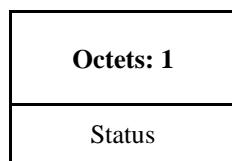
4521 Upon receipt, the Local Device shall determine whether the response status indicates that the Remote De-
4522vice is not a Primary Cache Device as indicated by a NOT_SUPPORTED status. If a NOT_SUPPORTED
4523status is returned, the Local Device should re-perform discovery on the Primary Discovery Cache. If a
4524NOT_PERMITTED status is returned, the local device must first issue a Discovery_store_req with a re-
4525turned SUCCESS status. If an INSUFFICIENT_SPACE status is returned, the Local Device shall discon-
4526tinue upload of discovery information, issue a Remove_node_cache_req (citing the Local Device), and
4527cease attempts to upload discovery information to the Remote Device.

4528 If a SUCCESS status is returned, the Local Device should continue to upload its remaining discovery cache
4529information to the Remote Device via the Active_EP_store_req and Simple_Desc_store_req.

4530 **2.4.4.2.16 Active_EP_store_rsp**

4531 The Active_EP_store_rsp command (ClusterID=0x8019) shall be formatted as illustrated in Figure 2.78.

4532 **Figure 2.78 Format of the Active_EP_store_rsp Command Frame**



4534 Table 2.107 specifies the fields of the Active_EP_store_rsp command frame.

4535 **Table 2.107 Fields of the Active_EP_store_rsp Command**

Name	Type	Valid Range	Description
Status	Integer	SUCCESS, INSUFFICIENT_SPACE, NOT_PERMITTED or NOT_SUPPORTED	The status of the Active_EP_store_rsp command.

4536 **2.4.4.2.16.1 When Generated**

4537 The Active_EP_store_rsp is provided to notify a Local Device of the request status from a Primary Dis-
4538 covery Cache device. Included in the response is a status code to notify the Local Device whether the re-
4539 quest is successful (the Primary Cache Device has space to store the discovery cache data for the Local De-
4540 vice), the request is not supported (meaning the Remote Device is not a Primary Discovery Cache device),
4541 or insufficient space exists.

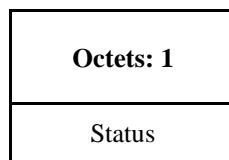
4542 **2.4.4.2.16.2 Effect on Receipt**

4543 Upon receipt, the Local Device shall determine whether the response status indicates that the Remote De-
4544 vice is not a Primary Cache Device as indicated by a NOT_SUPPORTED status. If a NOT_SUPPORTED
4545 status is returned, the Local Device should re-perform discovery on the Primary Discovery Cache. If a
4546 NOT_PERMITTED status is returned, the local device must first issue a Discovery_store_req with a re-
4547 turned SUCCESS status. If an INSUFFICIENT_SPACE status is returned, the Local Device shall discon-
4548 tinue upload of discovery information, issue a Remove_node_cache_req (citing the Local Device), and
4549 cease attempts to upload discovery information to the Remote Device. If a SUCCESS status is returned, the
4550 Local Device should continue to upload its remaining discovery cache information to the Remote Device
4551 via the Simple_Desc_store_req.

4552 **2.4.4.2.17 Simple_Desc_store_rsp**

4553 The Simple_Desc_store_rsp command (ClusterID=0x801a) shall be formatted as illustrated in Figure 2.79.

4554 **Figure 2.79 Format of the Simple_Desc_store_rsp Command Frame**



4555

4556 Table 2.108 specifies the fields of the Simple_Desc_store_rsp command frame.

4557 **Table 2.108 Fields of the Simple_Desc_store_rsp Command**

Name	Type	Valid Range	Description
Status	Integer	SUCCESS, INSUFFICIENT_SPACE, NOT_PERMITTED or NOT_SUPPORTED	The status of the Simple_desc_store_rsp command.

4558 **2.4.4.2.17.1 When Generated**

4559 The Simple_Desc_store_rsp is provided to notify a Local Device of the request status from a Primary Dis-
4560 covery Cache device. Included in the response is a status code to notify the Local Device whether the re-
4561 quest is successful (the Primary Cache Device has space to store the discovery cache data for the Local De-
4562 vice), the request is not supported (meaning the Remote Device is not a Primary Discovery Cache device),
4563 or insufficient space exists.

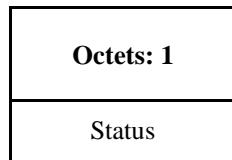
4564 **2.4.4.2.17.2 Effect on Receipt**

4565 Upon receipt, the Local Device shall determine whether the response status indicates that the Remote De-
4566 vice is not a Primary Cache Device as indicated by a NOT_SUPPORTED status. If a NOT_SUPPORTED
4567 status is returned, the Local Device should re-perform discovery on the Primary Discovery Cache. If a
4568 NOT_PERMITTED status is returned, the local device must first issue a Discovery_store_req with a re-
4569 turned SUCCESS status. If an INSUFFICIENT_SPACE status is returned, the Local Device shall discon-
4570 tinue upload of discovery information, issue a Remove_node_cache_req (citing the Local Device), and
4571 cease attempts to upload discovery information to the Remote Device. If a SUCCESS status is returned, the
4572 Local Device should continue to upload its remaining discovery cache information to the Remote Device
4573 via the Simple_Desc_store_req for other endpoints on the Local Device.

4574 **2.4.4.2.18 Remove_node_cache_rsp**

4575 The Remove_node_cache_rsp command (ClusterID=0x801b) shall be formatted as illustrated in Figure
4576 2.80.

4577 **Figure 2.80 Format of the Remove_node_cache_rsp Command Frame**



4578

4579 Table 2.109 specifies the fields of the Remove_node_cache_rsp command frame.

4580 **Table 2.109 Fields of the Remove_node_cache_rsp Command**

Name	Type	Valid Range	Description
Status	Integer	SUCCESS, DEVICE_NOT_FOUND or NOT_SUPPORTED	The status of the Remove_node_cache_rsp command

4581 **2.4.4.2.18.1 When Generated**

4582 The Remove_node_cache_rsp is provided to notify a Local Device of the request status from a Primary Dis-
4583 covery Cache device. Included in the response is a status code to notify the Local Device whether the re-
4584 quest is successful (the Primary Cache Device has removed the discovery cache data for the indicated de-
4585 vice of interest), or the request is not supported (meaning the Remote Device is not a Primary Discovery
4586 Cache device).

4587 **2.4.4.2.18.2 Effect on Receipt**

4588 Upon receipt, the Local Device shall determine whether the response status indicates that the Remote Device
4589 is not a Primary Cache Device as indicated by a NOT_SUPPORTED status. If a NOT_SUPPORTED status is returned,
4590 the Local Device should re-perform Find_node_cache_req to locate the Primary Discovery Cache device holding
4591 the discovery cache information for the indicated device of interest. When the Primary Discovery Cache device holding
4592 the discovery information for the device of interest is located, the Local Device should repeat the Remove_node_cache_req to successfully remove the discovery information.
4593 If a status of DEVICE_NOT_FOUND is received, this indicates that the Remote Device is the Primary Discovery Cache but does not hold the discovery information for the NWKAddr and the IEEEAddr presented in the request. The Local Device should employ the device discovery commands
4594 NWK_Addr_req and IEEE_Addr_req to determine the correct values for NWKAddr and IEEEAddr. If a
4595 SUCCESS status is returned, the Local Device has successfully removed the discovery cache information
4596 for the indicated device of interest within the request.
4597
4598
4599

4600 **2.4.4.2.19 Find_node_cache_rsp**

4601 The Find_node_cache_rsp command (ClusterID=0x801c) shall be formatted as illustrated in Figure 2.81.

4602 **Figure 2.81 Format of the Find_node_cache_rsp Command Frame**

Octets: 2	2	8
CacheNWKAddress	NWKAddr	IEEEAddr

4603
4604 Table 2.110 specifies the fields of the Find_node_cache_rsp command frame.
4605

Table 2.110 Fields of the Find_node_cache_rsp Command

Name	Type	Valid Range	Description
CacheNWKAddr	Device Address	16-bit NWK Address	NWK Address for the Primary Discovery Cache device holding the discovery information (or the device of interest if it responded to the request directly).
NWKAddr	Device Address	16-bit NWK Address	NWK Address for the device of interest.
IEEEAddr	Device Address	64-bit IEEE Address	IEEE address for the device of interest.

4606 **2.4.4.2.19.1 When Generated**

4607 The Find_node_cache_rsp is provided to notify a Local Device of the successful discovery of the Primary
4608 Discovery Cache device for the given NWKAddr and IEEEAddr fields supplied in the request, or to signify
4609 that the device of interest is capable of responding to discovery requests. The Find_node_cache_rsp shall
4610 be generated only by Primary Discovery Cache devices holding discovery information for the NWKAddr
4611 and IEEEAddr in the request or the device of interest itself and all other Remote Devices shall not supply a
4612 response.

4613 **2.4.4.2.19.2 Effect on Receipt**

4614 Upon receipt, the Local Device shall utilize the CacheNWKAddr as the Remote Device address for subsequent discovery requests relative to the NWKAddr and IEEEAddr in the response.

4616 **2.4.4.2.20 Extended_Simple_Desc_rsp**

4617 The Extended_Simple_Desc_rsp command (ClusterID=0x801d) shall be formatted as illustrated in Figure 4618 2.82.

4619 **Figure 2.82 Format of the Extended_Simple_Desc_rsp Command Frame**

Octet:1	2	1	1	1	1	Variable
Status	NWKAddr OfInterest	Endpoint	AppInput ClusterCount	AppOutput ClusterCount	startIndex	AppClusterList

4620

4621 Table 2.111 specifies the fields of the Extended_Simple_Desc_rsp command frame.

4622 **Table 2.111 Fields of the Extended_Simple_Desc_rsp Command**

Name	Type	Valid Range	Description
Status	Integer	SUCCESS, INVALID_EP, NOT_ACTIVE, DEVICE_NOT_FOUND, INV_REQUESTTYPE or NO_DESCRIPTOR	The status of the Extended_Simple_Desc_req command.
NWKAddrOfInterest	Device Address	16-bit NWK address	NWK address for the request.
Endpoint	8 bits	1-254	The endpoint on the destination.
AppInputClusterCount	8 bits	0x00-0xff	The total count of application input clusters in the Simple Descriptor for this endpoint.
AppOutputClusterCount	8 bits	0x00-0xff	The total count of application output clusters in the Simple Descriptor for this endpoint.
startIndex	8 bits	0x00-0xff	Starting index within the AppClusterList of the response represented by an ordered list of the Application Input Cluster List and Application Output Cluster List from the Simple Descriptor for this endpoint.

Name	Type	Valid Range	Description
AppClusterList			A concatenated, ordered list of the AppInputClusterList and AppOutputClusterList, beginning with StartIndex, from the Simple Descriptor. This field shall only be included in the frame if the status field is equal to SUCCESS.

4623 **2.4.4.2.20.1 When Generated**

4624 The Extended_Simple_Desc_rsp is generated by a remote device in response to an Extended_Simple_Desc_req directed to the remote device. This command shall be unicast to the originator of the
4625 Extended_Simple_Desc_req command.
4626

4627 The remote device shall generate the Extended_Simple_Desc_rsp command using the format illustrated in
4628 Table 2.111. The NWKAddrOfInterest field shall match that specified in the original Extended_Simple_Desc_req
4629 command. If the endpoint field specified in the original Extended_Simple_Desc_req
4630 command does not fall within the correct range specified in Table 2.50, the remote device shall set the Status
4631 field to INVALID_EP, set the Endpoint and StartIndex fields to their respective values supplied in the
4632 request, and not include the AppClusterList field.

4633 If the NWKAddrOfInterest field matches the network address of the remote device, it shall determine
4634 whether the endpoint field specifies the identifier of an active endpoint on the device. If the endpoint field
4635 corresponds to an active endpoint, the remote device shall set the Status field to SUCCESS, set the AppClusterList
4636 field to the sequence of octets from the concatenated AppInput ClusterList and AppOutput-
4637 ClusterList from the Simple Descriptor (see clause 2.3.2.3), and supply that field as AppClusterList in the
4638 response. Note that dependent on the value of StartIndex in the request, the results in AppClusterList may
4639 be empty (for example, the StartIndex begins after the sequence of octets given by the concatenation of
4640 AppInputClusterList and
4641 AppOutputClusterList). If the endpoint field does not correspond to an active endpoint, the remote device
4642 shall set the Status field to NOT_ACTIVE, set the StartIndex field to the value supplied in the request, and
4643 not include the AppClusterList field.

4644 **2.4.4.2.20.2 Effect on Receipt**

4645 On receipt of the Extended_Simple_Desc_rsp command, the recipient is either notified of the requested
4646 AppClusterList on the endpoint of the remote device indicated in the original Extended_Simple_Desc_req
4647 command or notified of an error. If the Extended_Simple_Desc_rsp command is received with a Status of
4648 SUCCESS, the AppClusterList field shall contain the requested portion of the application input cluster list
4649 and application output cluster list, starting with the StartIndex. Otherwise, the Status field indicates the er-
4650 rror and the AppClusterList field shall not be included.

4651 **2.4.4.2.21 Extended_Active_EP_rsp**

4652 The Extended_Active_EP_rsp command (ClusterID=0x801e) shall be formatted as illustrated in Figure
4653 2.83.

4654 **Figure 2.83 Format of the Extended_Active_EP_rsp Command Frame**

Octet: 1	2	1	1	Variable
Status	NWKAddrOfInterest	ActiveEPCount	StartIndex	ActiveEPList

4655

4656 Table 2.112 specifies the fields of the Extended_Active_EP_rsp command frame.

4657 **Table 2.112 Fields of the Extended_Active_EP_rsp Command**

Name	Type	Valid Range	Description
Status	Integer	SUCCESS, DEVICE_NOT_FOUND, INV_REQUESTTYPE or NO_DESCRIPTOR	The status of the Extended_Active_EP_req command.
NWKAddrOfInterest	Device Address	16-bit NWK address	NWK address for the request.
ActiveEPCount	Integer	0x00-0xff	The count of active endpoints on the Remote Device.
startIndex	Integer	0x00-0xff	Starting index for the list of active end- points for this report.
ActiveEPList			List of bytes each of which represents an 8-bit endpoint. The list begins with the entry starting with startIndex and continues until the remaining active endpoints are listed or the ASDU size is exhausted with whole endpoint entries.

4658

2.4.4.2.21.1 When Generated

4659
4660
4661

The Extended_Active_EP_rsp is generated by a remote device in response to an Extended_Active_EP_req directed to the remote device. This command shall be unicast to the originator of the Extended_Active_EP_req command.

4662
4663
4664
4665
4666
4667
4668
4669

The remote device shall generate the Extended_Active_EP_rsp command using the format illustrated in Table 2.111. The NWKAddrOfInterest field shall match that specified in the original Extended_Active_EP_req command. If the NWKAddrOfInterest field matches the network address of the remote device, it shall set the Status field to SUCCESS, set the ActiveEPCount field to the number of active endpoints on that device, and include an ascending list of all the identifiers of the active endpoints, beginning with startIndex, on that device in the ActiveEPList field and continuing until the remaining list of active endpoints from startIndex forward is listed or until the ASDU size is exhausted with whole endpoint entries.

If the NWKAddrOfInterest field does not match the network address of the remote device and it is an end device, it shall set the Status field to INV_REQUESTTYPE, set the ActiveEPCount field to 0, and not include the ActiveEPList field. If the NWKAddrOfInterest field does not match the network address of the remote device and it is the coordinator or a router, it shall determine whether the NWKAddrOfInterest field matches the network address of a device it holds in a discovery cache. If the NWKAddrOfInterest field does not match the network address of a device it holds in a discovery cache, it shall set the Status field to DEVICE_NOT_FOUND, set the ActiveEPCount field to 0, and not include the ActiveEPList field. If the NWKAddrOfInterest matches the network address of a device held in a discovery cache on the remote device, it shall determine whether that device has any active endpoints. If the discovery information corresponding to the ActiveEP request has not yet been uploaded to the discovery cache, the remote device shall set the Status field to NO_DESCRIPTOR, set the ActiveEPCount field to 0, and not include the ActiveEPList field. If the cached device has no active endpoints, the remote device shall set the Status field to SUCCESS, set the ActiveEPCount field to 0, and not include the ActiveEPList field. If the cached device has active endpoints, the remote device shall set the Status field to SUCCESS, set the ActiveEPCount field to the number of active endpoints on that device and include an ascending list of all the identifiers of the active endpoints, beginning with StartIndex, on that device in the ActiveEPList field.

4686 **2.4.4.2.21.2 Effect on Receipt**

4687 On receipt of the Extended_Active_EP_rsp command, the recipient is either notified of the active endpoints
4688 of the remote device indicated in the original Extended_Active_EP_req command or notified of an error. If
4689 the Extended_Active_EP_rsp command is received with a Status of SUCCESS, the ActiveEPCount field
4690 indicates the number of entries in the ActiveEPList field. Otherwise, the Status field indicates the error and
4691 the ActiveEPList field shall not be included. The requesting device may need to employ
4692 Extended_Active_EP_req multiple times, with different StartIndex values, to receive the full ActiveEPList
4693 from the remote device.

4694 **2.4.4.2.22 Parent_ance_rsp**

4695 The Parent_ance_rsp command (ClusterID = 0x801f) shall be formatted as illustrated in Figure 2.84, and
4696 is generated in response to a Parent_ance.
4697

4698 **Figure 2.84 Format of the Parent_ance_rsp Command Frame**

Octets: 1	1	Variable	...	Variable
Status	NumberOfChildren	ChildInfo[0]	...	ChildInfo[n]

4699 Table 2.113 specifies the fields of the Parent_ance_rsp.

4700 **Table 2.113 Fields of the Parent_ance_rsp**

Name	Type	Valid Range	Description
Status	Integer	SUCCESS, NOT_SUPPORTED, NO_MATCH	The status of the Parent_ance command.
NumberOfChildren	Integer	0-255	The number of ChildInfo structures contained in the message.
ChildInfo	ChildInfo	Variable	The child information. See Table 2.57.

4701
4702 Table 2.57 specifies the contents of the ChildInfo structure. This is the same format as the Parent_ance.
4703

4704 **2.4.4.2.22.1 When Generated**

4705 Upon receipt of a Parent_ance message, a router shall construct but not yet send a Parent_ance_rsp mes-
4706 sage with the NumberOfChildren field set to 0. It shall then examine each Extended Address present in
4707 the Parent_ance message and search its Neighbor Table for an entry that matches. If a device is found and
4708 the Device Type is ZigBee end device (0x02), the router shall do the following.

- 4709 1. If the Keepalive Received value is TRUE, it shall keep the parent/child relationship in the neigh-
4710 bor table unmodified. It shall then do the following:
4711 a. Append the ChildInfo structure to the Parent_ance_rsp.
4712 b. Increment NumberOfChildren by 1.
4713 2. If the Keepalive Received value is FALSE, it shall remove the entry.

4714 If the NumberOfChildren field value is 0, the local device shall discard the previously constructed Par-
4715 ent_Annce_rsp. No response message shall be sent.

4716 If the NumberOfChildren field in the Parent_Annce_rsp is greater than 0, it shall unicast the message to the
4717 sender of the Parent_Annce message.

4718 If the device has more ChildInfo entries than fit in a single message, it shall send additional messages.
4719 These messages do not have to be jittered or delayed since they are unicast to a single device. Each Par-
4720 ent_ance_rsp shall set the NumberOfChildren field to the number of entries contained within the message.

4721 **2.4.4.2.22.2 Effect on Receipt**

4722 On receipt of a Parent_ance_rsp, the device shall examine its Neighbor Table for each extended address in
4723 the ChildInfo entry and do the following.

- 4724 i) If the entry matches and the Device Type is Zigbee End Device (0x02), it shall do the following:
4725 (1) Delete the entry from the Neigbor table.
4726 ii) If the entry does not match, no more processing is performed on this ChildInfo entry.

4727 There is no message generated in response to a Parent_ance_rsp.

4728

4729 **2.4.4.3 End Device Bind, Bind, Unbind Bind Management Server
4730 Services**

4731 Table 2.114 lists the commands supported by Device Profile: End Device Bind, Bind and Unbind Server
4732 Services. Each of these primitives will be discussed in the following sections.

4733 **Table 2.114 End Device Bind, Unbind and Bind Management Server Services Primitives**

End Device Bind, Bind and Unbind Server Service Commands	Server Processing	Server Generation
End_Device_Bind_rsp	O	M
Bind_rsp	O	M

End Device Bind, Bind and Unbind Server Service Commands	Server Processing	Server Generation
Unbind_rsp	O	M
Bind_Register_rsp	O	M
Replace_Device_rsp	O	M
Store_Bkup_Bind_Entry_rsp	O	M
Remove_Bkup_Bind_Entry_rsp	O	M
Backup_Bind_Table_rsp	O	M
Recover_Bind_Table_rsp	O	M
Backup_Source_Bind_rsp	O	M
Recover_Source_Bind_rsp	O	M

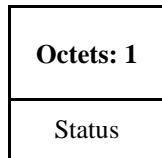
4734 For Server Generation requirements see section 2.4.4.1.

4735

2.4.4.3.1 End_Device_Bind_rsp

4737 The End_Device_Bind_rsp command (ClusterID=0x8020) shall be formatted as illustrated in Figure 2.85.

4738 **Figure 2.85 Format of the End_Device_Bind_rsp Command Frame**



4739

4740 Table 2.115 specifies the fields of the End_Device_Bind_rsp command frame.

4741 **Table 2.115 Fields of the End_Device_Bind_rsp Command**

Name	Type	Valid Range	Description
Status	Integer	SUCCESS, NOT_SUPPORTED, INVALID_EP, TIMEOUT, NO_MATCH, or DEVICE_BINDING_TABLE_FULL	The status of the End_Device_Bind_req command

4742 **2.4.4.3.1.1 When Generated**

4743 The End_Device_Bind_rsp is generated by the ZigBee Coordinator in response to an
4744 End_Device_Bind_req and contains the status of the request. This command shall be unicast to each device
4745 involved in the bind attempt, using the acknowledged data service.

4746 A Status of NOT_SUPPORTED indicates that the request was directed to a device which was not the
4747 ZigBee Coordinator or that the ZigBee Coordinator does not support End Device Binding. Otherwise,
4748 End_Device_Bind_req processing is performed as described below, including transmission of the
4749 End_Device_Bind_rsp.

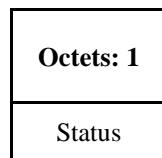
4750 **2.4.4.3.1.2 Effect on Receipt**

4751 When an End_Device_Bind_req is received, determination is made if a Status of NOT_SUPPORTED is
4752 warranted as indicated in the previous section. Assuming this device is the ZigBee Coordinator, the sup-
4753 plied endpoint shall be checked to determine whether it falls within the specified range. If it does not, a
4754 Status of INVALID_EP shall be returned. If the supplied endpoint falls within the specified range and if
4755 this is the first End_Device_Bind_req submitted for evaluation, it shall be stored and a timer started which
4756 expires at a pre-configured timeout value. This timeout value shall be a configurable item on the ZigBee
4757 Coordinator. If the timer expires before a second End_Device_Bind_req is received, a Status of TIMEOUT
4758 is returned. Otherwise, if a second End_Device_Bind_req is received within the timeout window, the two
4759 End_Device_Bind_req's are compared for a match. A Status of NO_MATCH indicates that two
4760 End_Device_Bind_req were evaluated for a match, but either the ProfileID parameters did not match (see
4761 section 2.3.3.2.2) or the ProfileID parameter matched but there was no match of any element of the InClus-
4762 terList or OutClusterList. A Status of SUCCESS means that a match was detected and a resulting Bind_req
4763 will subsequently be directed to the device indicated by the BindingTarget field of the
4764 End_Device_Bind_req with matched elements of the OutClusterList.

4765 **2.4.4.3.2 Bind_rsp**

4766 The Bind_rsp command (ClusterID=0x8021) shall be formatted as illustrated in Figure 2.86.

4767 **Figure 2.86 Format of the Bind_rsp Command Frame**



4768

4769 Table 2.116 specifies the fields of the Bind_rsp command frame.

4770 **Table 2.116 Fields of the Bind_rsp Command**

Name	Type	Valid Range	Description
Status	Integer	SUCCESS, NOT_SUPPORTED, INVALID_EP, TABLE_FULL or NOT_AUTHORIZED	The status of the Bind_req command.

4771 **2.4.4.3.2.1 When Generated**

4772 The Bind_rsp is generated in response to a Bind_req. If the Bind_req is processed and the Binding Table
4773 entry committed on the Remote Device, a Status of SUCCESS is returned. If the Remote Device is not a
4774 Primary binding table cache or the SrcAddress, a Status of NOT_SUPPORTED is returned. The Simple
4775 Descriptor in the receiving device correlating to the endpoint in the Bind_req shall be looked up. If the
4776 Simple Descriptor cannot be found then INVALID_EP shall be returned. If the Simple Descriptor is
4777 found, it shall be examined to see if the value of the ClusterID field in the Bind_Req message can be found
4778 within the Application output cluster list of the Simple Descriptor. If it cannot be found, then INVA-
4779 LID_EP shall be returned. If the Remote Device is the Primary binding table cache or SrcAddress but
4780 does not have Binding Table resources for the request, a Status of TABLE_FULL is returned.

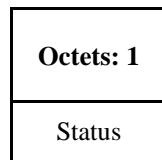
4781 **2.4.4.3.2.2 Effect on Receipt**

4782 Upon receipt, error checking is performed on the request as described in the previous section. Assuming the
4783 Status is SUCCESS, the parameters from the Bind_req are entered into the Binding Table at the Remote
4784 Device via the APSME-BIND.request primitive.

4785 **2.4.4.3.3 Unbind_rsp**

4786 The Unbind_rsp command (ClusterID=0x8022) shall be formatted as illustrated in Figure 2.87.

4787 **Figure 2.87 Format of the Unbind_rsp Command Frame**



4788

4789 Table 2.117 specifies the fields of the Unbind_rsp command frame.

4790 **Table 2.117 Fields of the Unbind_rsp Command**

Name	Type	Valid Range	Description
Status	Integer	SUCCESS, NOT_SUPPORTED, INVALID_EP, NO_ENTRY or NOT_AUTHORIZED	The status of the Unbind_req command.

4791 **2.4.4.3.3.1 When Generated**

4792 The Unbind_rsp is generated in response to an Unbind_req. If the Unbind_req is processed and the corre-
4793 sponding Binding Table entry is removed from the Remote Device, a Status of SUCCESS is returned. If the
4794 Remote Device is not the ZigBee Coordinator or the SrcAddress, a Status of NOT_SUPPORTED is re-
4795 turned. The supplied endpoint shall be checked to determine whether it falls within the specified range. If it
4796 does not, a Status of INVALID_EP shall be returned. If the Remote Device is the ZigBee Coordinator or
4797 SrcAddress but does not have a Binding Table entry corresponding to the parameters received in the re-
4798 quest, a Status of NO_ENTRY is returned.

4799 **2.4.4.3.3.2 Effect on Receipt**

4800 Upon receipt, error checking is performed on the response. If the status is SUCCESS, the device has suc-
4801 cessfully removed the binding entry for the parameters specified in the Unbind_req.

4802 **2.4.4.3.4 Bind_Register_rsp**

4803 The Bind_Register_rsp command (ClusterID=0x8023) shall be formatted as illustrated in Figure 2.88.

4804 **Figure 2.88 Format of the Bind_Register_rsp Command Frame**

Octets: 1	2	2	Variable
Status	BindingTableEntries	BindingTableListCount	BindingTableList

4805

4806 Table 2.118 specifies the fields of the Bind_Register_rsp command frame.

4807 **Table 2.118 Fields of the Bind_Register_rsp Command**

Name	Type	Valid Range	Description
Status	Integer	SUCCESS, NOT_SUPPORTED, TABLE_FULL	The status of the Bind_Register_reg command.
BindingTableEntries	Integer	0x0000 - 0ffff	Number of binding table entries for the requesting device held by the primary binding table cache.
BindingTableListCount	Integer	0x0000 - 0xffff	Number of source binding table entries contained in this response.
BindingTableList	List of source binding descriptors	This list shall contain the number of elements given by the BindingTableList-Count	A list of source binding.

4808 **2.4.4.3.4.1 When Generated**

4809 The Bind_Register_rsp is generated from a primary binding table cache device in response to a
4810 Bind_Register_req and contains the status of the request. This command shall be unicast to the requesting
4811 device.

4812 If the device receiving the Bind_Register_req is not a primary binding table cache a Status of
4813 NOT_SUPPORTED is returned. If its list of devices which choose to store their own binding table entries
4814 is full, a status of TABLE_FULL is returned. In these error cases, BindingTableEntries and BindingTable-
4815 ListCount shall be zero and BindingTableList shall be empty. A Status of SUCCESS indicates that the re-
4816 questing device has been successfully registered.

4817 In the successful case, the primary binding table cache device shall search its cache for existing entries
4818 whose source address is the same as the parameter supplied in the Bind_Register_req command. The num-
4819 ber of such entries is given in the response as BindingTableEntries. The entries are used to generate Bind-
4820 ingTableList up to the maximum that can be contained in the response. The actual number of entries is
4821 given in the response as BindingTableListCount and may be less than BindingTableEntries if this is too
4822 large. In this case (which is expected to be rare) the primary binding table cache device shall use Bind_req
4823 commands to send the rest of the entries to the requesting device.

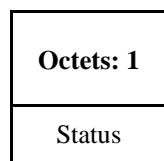
4824 **2.4.4.3.4.2 Effect on Receipt**

4825 The requesting device is notified of the results of its attempt to register. If successful, it shall store the
4826 source binding table entries from the response into its source binding table.

4827 **2.4.4.3.5 Replace_Device_rsp**

4828 The Replace_Device_rsp command (ClusterID=0x8024) shall be formatted as illustrated in Figure 2.89.

4829 **Figure 2.89 Format of the Replace_Device_rsp Command Frame**



4830

4831 Table 2.119 specifies the fields of the Replace_Device_rsp command frame.

4832 **Table 2.119 Fields of the Replace_Device_rsp Command**

Name	Type	Valid Range	Description
Status	Integer	NOT_SUPPORTED, INV_REQUESTTYPE	The status of the Replace_Device_req command.

4833 **2.4.4.3.5.1 When Generated**

4834 The Replace_Device_rsp is generated from a primary binding table cache device in response to a Replace_Device_req and contains the status of the request. This command shall be unicast to the requesting device. If the device receiving the Replace_Device_req is not a primary binding table cache, a Status of NOT_SUPPORTED is returned. The primary binding table cache shall search its binding table for entries whose source address and source endpoint, or whose destination address and destination endpoint match OldAddress and OldEndpoint, as described in the text for Replace_Device_req. It shall change these entries to have NewAddress and possibly NewEndpoint. It shall then return a response of SUCCESS.

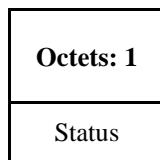
4841 **2.4.4.3.5.2 Effect on Receipt**

4842 The requesting device is notified of the status of its Replace_Device_req command.

4843 **2.4.4.3.6 Store_Bkup_Bind_Entry_rsp**

4844 The Store_Bkup_Bind_Entry_rsp command (ClusterID=0x8025) shall be formatted as illustrated in Figure 4845 2.90.

4846 **Figure 2.90 Format of the Store_Bkup_Bind_Entry_rsp Command Frame**



4847

4848 Table 2.120 specifies the fields of the Store_Bkup_Bind_Entry_rsp command frame.

4849 **Table 2.120 Fields of the Store_Bkup_Bind_Entry_rsp Command**

Name	Type	Valid Range	Description
Status	Integer	SUCCESS, NOT_SUPPORTED, INV_REQUESTTYPE. TABLE_FULL	The status of the Store_Bkup_Bind_Entry_rsp command.

4850 **2.4.4.3.6.1 When Generated**

4851 The Store_Bkup_Bind_Entry_rsp is generated from a backup binding table cache device in response to a
4852 Store_Bkup_Bind_Entry_req from a primary binding table cache, and contains the status of the request.
4853 This command shall be unicast to the requesting device. If the remote device is not a backup binding table
4854 cache, it shall return a status of NOT_SUPPORTED. If the originator of the request is not recognized as a
4855 primary binding table cache, it shall return a status of INV_REQUESTTYPE. Otherwise, the backup binding
4856 table cache shall add the binding entry to its binding table and return a status of SUCCESS. If there is
4857 no room, it shall return a status of TABLE_FULL.

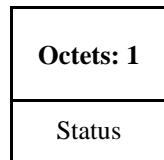
4858 **2.4.4.3.6.2 Effect on Receipt**

4859 The requesting device is notified of the status of its attempt to store a bind entry.

4860 **2.4.4.3.7 Remove_Bkup_Bind_Entry_rsp**

4861 The Remove_Bkup_Bind_Entry_rsp command (ClusterID=0x8026) shall be formatted as illustrated in
4862 Figure 2.91.

4863 **Figure 2.91 Format of the Remove_Bkup_Bind_Entry_rsp Command Frame**



4864

4865 Table 2.121 specifies the fields of the Remove_Bkup_Bind_Entry_rsp command frame.

4866 **Table 2.121 Fields of the Remove_Bkup_Bind_Entry_rsp Command**

Name	Type	Valid Range	Description
Status	Integer	SUCCESS, NOT_SUPPORTED, INV_REQUESTTYPE, NO_ENTRY	The status of the Remove_Bkup_Bind_Entry_rsp command.

4867 **2.4.4.3.7.1 When Generated**

4868 The Remove_Bkup_Bind_Entry_rsp is generated from a backup binding table cache device in response to a
4869 Remove_Bkup_Bind_Entry_req from the primary binding table cache and contains the status of the
4870 request. This command shall be unicast to the requesting device. If the remote device is not a backup binding
4871 table cache, it shall return a status of NOT_SUPPORTED. If the originator of the request is not recognized
4872 as a primary binding table cache, it shall return a status of INV_REQUESTTYPE. Otherwise, the backup binding
4873 table cache shall delete the binding entry from its binding table and return a status of SUCCESS. If
4874 the entry is not found, it shall return a status of NO_ENTRY.

4875 **2.4.4.3.7.2 Effect on Receipt**

4876 The requesting device is notified of the status of its attempt to remove a bind entry from the backup cache.

4877 **2.4.4.3.8 Backup_Bind_Table_rsp**

4878 The Backup_Bind_Table_rsp command (ClusterID=0x8027) shall be formatted as illustrated in Figure
4879 2.92.

4880

Figure 2.92 Format of the Backup_Bind_Table_rsp Command Frame

Octets: 1	2
Status	EntryCount

4881

4882 Table 2.122 specifies the fields of the Backup_Bind_Table_rsp command frame.

4883

Table 2.122 Fields of the Backup_Bind_Table_rsp Command

Name	Type	Valid Range	Description
Status	Integer	SUCCESS, NOT_SUPPORTED, TABLE_FULL, INV_REQUESTTYPE	The status of the Backup_Bind_Table_rsp command.
EntryCount	Integer	0x0000 - 0xFFFF	The number of entries in the backup binding table.

4884

2.4.4.3.8.1 When Generated

4885

The Backup_Bind_Table_rsp is generated from a backup binding table cache device in response to a Backup_Bind_Table_req from a primary binding table cache and contains the status of the request. This command shall be unicast to the requesting device. If the remote device is not a backup binding table cache, it shall return a status of NOT_SUPPORTED. If the originator of the request is not recognized as a primary binding table cache, it shall return a status of INV_REQUESTTYPE. Otherwise, the backup binding table cache shall overwrite the binding entries in its binding table starting with StartIndex and continuing for

4892

BindingTableListCount entries. If this exceeds its table size, it shall fill in as many entries as possible and return a status of TABLE_FULL and the EntryCount parameter will be the number of entries in the table. Otherwise, it shall return a status of SUCCESS and EntryCount will be equal to StartIndex + BindingTableListCount from Backup_Bind_Table_req.

4896

2.4.4.3.8.2 Effect on Receipt

4897

The requesting device is notified of the status of its attempt to store a binding table.

4898

2.4.4.3.9 Recover_Bind_Table_rsp

4899

The Backup_Bind_Table_rsp command (ClusterID=0x8028) shall be formatted as illustrated in Figure 2.93.

4901

Figure 2.93 Format of the Backup_Bind_Table_rsp Command Frame

Octets: 1	2	2	2	Variable
Status	BindingTableEntries	StartIndex	BindingTableListCount	BindingTableList

4902

4903

Table 2.123 specifies the fields of the Recover_Bind_Table_rsp command frame.

4904

Table 2.123 Fields of the Recover_Bind_Table_rsp Command

Name	Type	Valid Range	Description
Status	Integer	SUCCESS, NOT_SUPPORTED, INV_REQUESTTYPE, NO_ENTRY	The status of the Recover_Bind_Table_rsp command.
BindingTableEntries	Integer	0x0000 - 0xffff	Total number of binding table entries in the backup binding cache.
startIndex	Integer	0x0000 - 0xffff	Starting index within the binding table to begin reporting for the binding table list.
BindingTableListCount	Integer	0x0000 - 0xffff	Number of binding entries included within BindingTableList.
BindingTableList	Integer	The list shall contain the number of elements given by BindingTableListCount	A list of descriptors, beginning with the startIndex element and continuing for BindingTableListCount of elements in the backup binding table cache.

4905

2.4.4.3.9.1 When Generated

4906 The Recover_Bind_Table_rsp is generated from a backup binding table cache device in response to a
 4907 Recover_Bind_Table_req from a primary binding table cache and contains the status of the request. This
 4908 command shall be unicast to the requesting device. If the responding device is not a backup binding table
 4909 cache, it shall return a status of NOT_SUPPORTED. If the originator of the request is not recognized as a
 4910 primary binding table cache it shall return a status of INV_REQUESTTYPE. Otherwise, the backup binding
 4911 table cache shall prepare a list of binding table entries from its backup beginning with startIndex. It will
 4912 fit in as many entries as possible into a Recover_Bind_Table_rsp command and return a status of SUC-
 4913 CESS. If startIndex is more than the number of entries in the Binding table, a status of NO_ENTRY is re-
 4914 turned. For a successful response, BindingTableEntries is the total number of entries in the backup binding
 4915 table, and BindingTableListCount is the number of entries which is being returned in the response.

4916

2.4.4.3.9.2 Effect on Receipt

4917

The requesting device is notified of the status of its attempt to restore a binding table.

4918

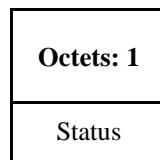
2.4.4.3.10 Backup_Source_Bind_rsp

4919
4920

The Backup_Source_Bind_rsp command (ClusterID=0x8029) shall be formatted as illustrated in Figure
2.94.

4921

Figure 2.94 Format of the Backup_Source_Bind_rsp Command Frame



4922

4923

Table 2.124 specifies the fields of the Backup_Source_Bind_rsp command frame.

4924

Table 2.124 Fields of the Backup_Source_Bind_rsp Command

Name	Type	Valid Range	Description
Status	Integer	SUCCESS, NOT_SUPPORTED, TABLE_FULL, INV_REQUESTTYPE	The status of the Back-up_Source_Bind_rsp command.

4925

2.4.4.3.10.1 When Generated

4926
4927
4928
4929
4930
4931
4932
4933

The Backup_Source_Bind_rsp is generated from a backup binding table cache device in response to a Backup_Source_Bind_req from a primary binding table cache and contains the status of the request. This command shall be unicast to the requesting device. If the remote device is not a backup binding table cache, it shall return a status of NOT_SUPPORTED. If the originator of the request is not recognized as a primary binding table cache, it shall return a status of INV_REQUESTTYPE. Otherwise, the backup binding table cache shall overwrite its backup source binding table starting with StartIndex and continuing for BindingTableListCount entries. If this exceeds its table size, it shall return a status of TABLE_FULL. Otherwise it shall return a status of SUCCESS.

4934

2.4.4.3.10.2 Effect on Receipt

4935

The requesting device is notified of the status of its attempt to backup the source binding table.

4936

2.4.4.3.11 Recover_Source_Bind_rsp

4937
4938

The Recover_Source_Bind_rsp command (ClusterID=0x802a) shall be formatted as illustrated in Figure 2.95.

4939

Figure 2.95 Format of the Recover_Source_Bind_rsp Command Frame

Octets: 1	2	2	2	Variable
Status	SourceTableEntries	StartIndex	SourceTableListCount	SourceTableList

4940

Table 2.125 specifies the fields of the Recover_Source_Bind_rsp command frame.

4941

Table 2.125 Fields of the Recover_Source_Bind_rsp Command

Name	Type	Valid Range	Description
Status	Integer	SUCCESS, NOT_SUPPORTED, TABLE_FULL, INV_REQUESTTYPE	The status of the Recover_Source_Bind_rsp command.
SourceTableEntries	Integer	0x0000 - 0xffff	Total number of source table entries in the backup binding cache.
startIndex	Integer	0x0000 - 0xffff	Starting index within the source table to begin reporting for the source table list.

Name	Type	Valid Range	Description
SourceTableListCount	Integer	0x0000 - 0xffff	Number of source table entries included within SourceTableList.
SourceTableList	List of source descriptors	The list shall contain the number of elements given by SourceTableListCount	A list of descriptors, beginning with the StartIndex element and continuing for SourceTableListCount of elements in the backup source table cache (consisting of IEEE addresses).

2.4.4.3.11.1 When Generated

The Recover_Source_Bind_rsp is generated from a backup binding table cache device in response to a Recover_Source_Bind_req from a primary binding table cache and contains the status of the request. This command shall be unicast to the requesting device. If the responding device is not a backup binding table cache, it shall return a status of NOT_SUPPORTED. If the originator of the request is not recognized as a primary binding table cache, it shall return a status of INV_REQUESTTYPE. Otherwise, the backup binding table cache shall prepare a list of binding table entries from its backup beginning with StartIndex. It will fit in as many entries as possible into a Recover_Source_Bind_rsp command and return a status of SUCCESS. If StartIndex is more than the number of entries in the Source table, a status of NO_ENTRY is returned. For a successful response, SourceTableEntries is the total number of entries in the backup source table, and SourceTableListCount is the number of entries which is being returned in the response.

2.4.4.3.11.2 Effect on Receipt

The requesting device is notified of the status of its attempt to restore a source binding table.

2.4.4.4 Network Management Server Services

Table 2.126 lists the commands supported by Device Profile: Network Management Server Services. Each of these commands will be discussed in the following sections.

Table 2.126 Network Management Server Service Commands

Network Management Server Service Command	Server Processing	Server Generation
Mgmt_NWK_Disc_rsp	O	M
Mgmt_Lqi_rsp	M ²	M
Mgmt_Rtg_rsp	O	M
Mgmt_Bind_rsp	O	M
Mgmt_Leave_rsp	O	M
Mgmt_Direct_Join_rsp	O	M

² CCB 1604

Network Management Server Service Command	Server Processing	Server Generation
Mgmt_Permit_Joining_rsp	M	M
Mgmt_Cache_rsp	O	M
Mgmt_NWK_Update_notify	O	M

4960 For Server Generation requirements see section 2.4.4.1.

4961

2.4.4.4.1 Mgmt_NWK_Disc_rsp

4963 The Mgmt_NWK_Disc_rsp command (ClusterID=0x8030) shall be formatted as illustrated in Figure 2.96.

4964 **Figure 2.96 Format of the Mgmt_NWK_Disc_rsp Command Frame**

Octets: 1	1	1	1	Variable
Status	NetworkCount	StartIndex	NetworkListCount	NetworkList

4965

4966 Table 2.127 specifies the fields of the Mgmt_NWK_Disc_rsp command frame.

4967

Table 2.127 Fields of the Mgmt_NWK_Disc_rsp Command

Name	Type	Valid Range	Description
Status	Integer	NOT_SUPPORTED or any status code returned from the NLME-NETWORK-DISCOVERY.req primitive.	The status of the Mgmt_NWK_Disc_req command.
NetworkCount	Integer	0x00-0xff	The total number of networks reported by the NLME-NETWORK-DISCOVERY.confirm.
StartIndex	Integer	0x00-0xff	The starting point in the NetworkList from the NLME-NETWORK-DISCOVERY.confirm where reporting begins for this response.
NetworkList-Count	Integer	0x00-0xff	The number of network list descriptors reported within this response.

Name	Type	Valid Range	Description
NetworkList	List of Network Descriptors	The list shall contain the number of elements given by the NetworkList-Count parameter.	A list of descriptors, one for each of the networks discovered, beginning with the StartIndex element and continuing for NetworkListCount, of the elements returned by the NLME-NETWORK-DISCOVERY.conf rm primitive. Each entry shall be formatted as illustrated in Table 2.128.

4968

4969

Table 2.128 NetworkList Record Format

Name	Size (Bits)	Valid Range	Description
ExtendedPanID	64	A 64-bit PAN identifier	The 64-bit extended PAN identifier of the discovered network.
LogicalChannel	8	Selected from the available logical channels supported by the PHY (see [B1])	The current logical channel occupied by the network.
StackProfile	4	0x0-0xf	A ZigBee stack profile identifier indicating the stack profile in use in the discovered network.
ZigBeeVersion	4	0x0-0xf	The version of the ZigBee protocol in use in the discovered network.
BeaconOrder	4	0x0-0xf	This specifies how often the MAC sub-layer beacon is to be transmitted by a given device on the network. For a discussion of MAC sub-layer beacon order see [B1].
SuperframeOrder	4	0x0-0xf	For beacon-oriented networks, <i>i.e.</i> , beacon order < 15, this specifies the length of the active period of the superframe. For a discussion of MAC sub-layer superframe order see [B1].
PermitJoining	1	TRUE or FALSE	A value of TRUE indicates that at least one ZigBee router on the network currently permits joining, <i>i.e.</i> , its NWK has been issued an NLME-PERMIT-JOINING primitive and the time limit, if given, has not yet expired.
Reserved	7		Each of these bits shall be set to 0.

4970 **2.4.4.4.1.1 When Generated**

4971 The Mgmt_NWK_Disc_rsp is generated in response to an Mgmt_NWK_Disc_req. If this management
4972 command is not supported, a status of NOT_SUPPORTED shall be returned and all parameter fields after
4973 the Status field shall be omitted. Otherwise, the Remote Device shall implement the following process.

4974 Upon receipt of and after support for the Mgmt_NWK_Disc_req has been verified, the Remote Device
4975 shall issue an NLME-NETWORK-DISCOVERY.request primitive using the ScanChannels and ScanDuration
4976 parameters, supplied in the Mgmt_NWK_Disc_req command. Upon receipt of the
4977 NLME-NETWORK-

4978 DISCOVERY.confirm primitive, the Remote Device shall report the results, starting with the StartIndex
4979 element, via the Mgmt_NWK_Disc_rsp command. The NetworkList field shall contain whole NetworkList
4980 records, formatted as specified in Table 2.128, until the limit on MSDU size, i.e., *aMaxMACFrameSize* (see
4981 [B1]), is reached. The number of results reported shall be set in the NetworkListCount.

4982 **2.4.4.4.1.2 Effect on Receipt**

4983 The local device is notified of the results of its attempt to perform a remote network discovery.

4984 **2.4.4.4.2 Mgmt_Lqi_rsp**

4985 The Mgmt_Lqi_rsp command (ClusterID=0x8031) shall be formatted as illustrated in Figure 2.97.

4986 **Figure 2.97 Format of the Mgmt_Lqi_rsp Command Frame**

Octets: 1	1	1	1	Variable
Status	NeighborTable Entries	Start Index	NeighborTable ListCount	NeighborTable List

4987
4988 Table 2.129 specifies the fields of the Mgmt_Lqi_rsp command frame.
4989

Table 2.129 Fields of the Mgmt_Lqi_rsp Command

Name	Type	Valid Range	Description
Status	Integer	NOT_SUPPORTED or any status code returned from the NLME-GET.confirm primitive	The status of the Mgmt_Lqi_req command.
NeighborTableEntries	Integer	0x00-0xff	Total number of Neighbor Table entries within the Remote Device.
startIndex	Integer	0x00-0xff	Starting index within the Neighbor Table to begin reporting for the NeighborTableList.
NeighborTableListCount	Integer	0x00-0x02	Number of Neighbor Table entries included within NeighborTableList.

NeighborTableList	List of Neighbor Descriptors	The list shall contain the number elements given by the NeighborTableListCount	A list of descriptors, beginning with the StartIndex element and continuing for NeighborTableListCount, of the elements in the Remote Device's Neighbor Table including the device address and associated LQI (see Table 2.130 for details).
-------------------	------------------------------	--	--

4990

4991

Table 2.130 NeighborTableList Record Format

Name	Size (Bits)	Valid Range	Description
Extended PAN Id	64	A 64-bit PAN identifier	The 64-bit extended PAN identifier of the neighboring device.
Extended address	64	An extended 64-bit, IEEE address	64-bit IEEE address that is unique to every device. If this value is unknown at the time of the request, this field shall be set to 0xfffffffffffffff.
Network address	16	Network address	The 16-bit network address of the neighboring device.
Device type	2	0x00 - 0x03	The type of the neighbor device: 0x00 = ZigBee coordinator 0x01 = ZigBee router 0x02 = ZigBee end device 0x03 = Unknown
RxOnWhenIdle	2	0x00 - 0x02	Indicates if neighbor's receiver is enabled during idle portions of the CAP: 0x00 = Receiver is off 0x01 = Receiver is on 0x02 = unknown
Relationship	3	0x00 - 0x04	The relationship between the neighbor and the current device: 0x00 = neighbor is the parent 0x01 = neighbor is a child 0x02 = neighbor is a sibling 0x03 = None of the above 0x04 = previous child

Name	Size (Bits)	Valid Range	Description
Reserved	1		This reserved bit shall be set to 0.
Permit joining	2	0x00 - 0x02	An indication of whether the neighbor device is accepting join requests: 0x00 = neighbor is not accepting join requests 0x01 = neighbor is accepting join requests 0x02 = unknown
Reserved	6		Each of these reserved bits shall be set to 0.
Depth	8	0x00 - nwkcMaxDepth	The tree depth of the neighbor device. A value of 0x00 indicates that the device is the ZigBee coordinator for the network.
LQI	8	0x00 - 0xff	The estimated link quality for RF transmissions from this device. See [B1] for discussion of how this is calculated.

4992 **2.4.4.4.2.1 When Generated**

4993 The Mgmt_Lqi_rsp is generated in response to an Mgmt_Lqi_req. If this management command is not
 4994 supported, a status of NOT_SUPPORTED shall be returned and all parameter fields after the Status field
 4995 shall be omitted. Otherwise, the Remote Device shall implement the following processing.

4996 Upon receipt of and after support for the Mgmt_Lqi_req has been verified, the Remote Device shall per-
 4997 form an NLME-GET.request (for the *nwkNeighborTable* attribute) and process the resulting neighbor table
 4998 (obtained via the NLME-GET.confirm primitive) to create the Mgmt_Lqi_rsp command. If *nwkNeigh-
 4999 borTable* was successfully obtained but one or more of the fields required in the NeighborTableList record
 5000 (see Table 2.130) are not supported (as they are optional), the Mgmt_Lqi_rsp shall return a status of
 5001 NOT_SUPPORTED and all parameter fields after the Status field shall be omitted. Otherwise, the
 5002 Mgmt_Lqi_rsp command shall contain the same status that was contained in the NLME-GET.confirm
 5003 primitive and if this was not SUCCESS, all parameter fields after the status field shall be omitted.

5004 From the *nwkNeighborTable* attribute, the neighbor table shall be accessed, starting with the index speci-
 5005 fied by StartIndex, and shall be moved to the NeighborTableList field of the Mgmt_Lqi_rsp command. The
 5006 entries reported from the neighbor table shall be those, starting with StartIndex and including whole
 5007 NeighborTableList records (see Table 2.130) until the limit on MSDU size, i.e., *aMaxMACFrameSize* (see
 5008 [B1]), is reached. Within the Mgmt_Lqi_Rsp command, the NeighborTableEntries field shall represent the
 5009 total number of Neighbor Table entries in the Remote Device. The parameter NeighborTableListCount
 5010 shall be the number of entries reported in the NeighborTableList field of the Mgmt_Lqi_rsp command.

5011 The extended address, device type, RxOnWhenIdle, and permit joining fields have “unknown” values
 5012 which shall be returned where the values are not available.

5013 **2.4.4.4.2.2 Effect on Receipt**

5014 The local device is notified of the results of its attempt to obtain the neighbor table.

5015 **2.4.4.4.3 Mgmt_Rtg_rsp**

5016 The Mgmt_Rtg_rsp command (ClusterID=0x8032) shall be formatted as illustrated in Figure 2.98.

5017 **Figure 2.98 Format of the Mgmt_Rtg_rsp Command Frame**

Octets: 1	1	1	1	Variable
Status	RoutingTable Entries	Start Index	RoutingTable ListCount	RoutingTable List

5018

5019 Table 2.131 specifies the fields of the Mgmt_Rtg_rsp command frame.

5020 **Table 2.131 Fields of the Mgmt_Rtg_rsp Command**

Name	Type	Valid Range	Description
Status	Integer	NOT_SUPPORTED or any status code returned from the NLME-GET.confirm primitive	The status of the Mgmt_Rtg_req command.
RoutingTableEntries	Integer	0x00-0xff	Total number of Routing Table entries within the Remote Device.
startIndex	Integer	0x00-0xff	Starting index within the Routing Table to begin reporting for the RoutingTableList.
RoutingTableListCount	Integer	0x00-0xff	Number of Routing Table entries included within RoutingTableList.
RoutingTableList	List of Routing Descriptors	The list shall contain the number elements given by the RoutingTableListCount	A list of descriptors, beginning with the startIndex element and continuing for RoutingTableListCount, of the elements in the Remote Device's Routing Table (see Table 2.132 for details).

5021

5022

Table 2.132 RoutingTableList Record Format

Name	Size (Bits)	Valid Range	Description
Destination address	16	The 16-bit network address of this route.	Destination address.
Status	3	The status of the route.	0x0=ACTIVE. 0x1=DISCOVERY_UNDERWAY. 0x2=DISCOVERY_FAILED. 0x3=INACTIVE. 0x4=VALIDATION_UNDERWAY 0x5-0x7=RESERVED.
Memory Constrained	1		A flag indicating whether the device is a memory constrained concentrator.
Many-to-one	1		A flag indicating that the destination is a concentrator that issued a many-to-one request.
Route record required	1		A flag indicating that a route record command frame should be sent to the destination prior to the next data packet.
Reserved	2		
Next-hop address	16	The 16-bit network address of the next hop on the way to the destination.	Next-hop address.

5023

2.4.4.4.3.1 When Generated

5024
5025
5026

The Mgmt_Rtg_rsp is generated in response to an Mgmt_Rtg_req. If this management command is not supported, a status of NOT_SUPPORTED shall be returned and all parameter fields after the Status field shall be omitted. Otherwise, the Remote Device shall implement the following processing.

5027
5028
5029
5030
5031

Upon receipt of and after support for the Mgmt_Rtg_req has been verified, the Remote Device shall perform an NLME-GET.request (for the *nwkRouteTable* attribute) and process the resulting NLME-GET.confirm (containing the *nwkRouteTable* attribute) to create the Mgmt_Rtg_rsp command. The Mgmt_Rtg_rsp command shall contain the same status that was contained in the NLME-GET.confirm primitive and if this was not SUCCESS, all parameter fields after the status field shall be omitted.

5032
5033
5034
5035
5036
5037
5038

From the *nwkRouteTable* attribute, the routing table shall be accessed, starting with the index specified by StartIndex, and moved to the RoutingTableList field of the Mgmt_Rtg_rsp command. The entries reported from the routing table shall be those, starting with StartIndex and including whole RoutingTableList records (see Table 2.132) until MSDU size limit, i.e., *aMaxMACFrameSize* (see [B1]), is reached. Within the Mgmt_Rtg_Rsp command, the RoutingTableEntries field shall represent the total number of Routing Table entries in the Remote Device. The RoutingTableListCount field shall be the number of entries reported in the RoutingTableList field of the Mgmt_Rtg_req command.

5039 **2.4.4.4.3.2 Effect on Receipt**

5040 The local device is notified of the results of its attempt to obtain the routing table.

5041 **2.4.4.4.4 Mgmt_Bind_rsp**

5042 The Mgmt_Bind_rsp command (ClusterID=0x8033) shall be formatted as illustrated in Figure 2.99.

5043 **Figure 2.99 Format of the Mgmt_Bind_rsp Command Frame**

Octets: 1	1	1	1	Variable
Status	BindingTable Entries	Start Index	BindingTable ListCount	BindingTable List

5044

5045 Table 2.133 specifies the fields of the Mgmt_Bind_rsp command frame.

5046 **Table 2.133 Fields of the Mgmt_Bind_rsp Command**

Name	Type	Valid Range	Description
Status	Integer	NOT_SUPPORTED or any status code returned from the APSME-GET.confirm primitive	The status of the Mgmt_Bind_req command.
BindingTableEntries	Integer	0x00-0xff	Total number of Binding Table entries within the Remote Device.
startIndex	Integer	0x00-0xff	Starting index within the Binding Table to begin reporting for the BindingTableList.
BindingTableListCount	Integer	0x00-0xff	Number of Binding Table entries included within BindingTableList.
BindingTableList	List of Binding Descriptors	The list shall contain the number elements given by the BindingTableListCount	A list of descriptors, beginning with the startIndex element and continuing for BindingTableListCount, of the elements in the Remote Device's Binding Table (see Table 2.134 for details).

5047

5048

Table 2.134 BindingTableList Record Format

Name	Size (Bits)	Valid Range	Description
SrcAddr	64	A valid 64-bit IEEE address	The source IEEE address for the binding entry.
SrcEndpoint	8	0x01 - 0xfe	The source endpoint for the binding entry.
ClusterId	16	0x0000 - 0xffff	The identifier of the cluster on the source device that is bound to the destination device.
DstAddrMode	8	0x00 - 0xff	The addressing mode for the destination address. This field can take one of the non-reserved values from the following list: 0x00 = reserved 0x01 = 16-bit group address for DstAddr and DstEndpoint not present 0x02 = reserved 0x03 = 64-bit extended address for DstAddr and DstEndp present 0x04 – 0xff = reserved
DstAddr	16/64	As specified by the DstAddr-Mode field	The destination address for the binding entry.
DstEndpoint	0/8	0x01 - 0xff	This field shall be present only if the DstAddrMode field has a value of 0x03 and, if present, shall be the destination endpoint for the binding entry.

5049

2.4.4.4.4.1 When Generated

5050
5051
5052

The Mgmt_Bind_rsp is generated in response to a Mgmt_Bind_req. If this management command is not supported, a status of NOT_SUPPORTED shall be returned and all parameter fields after the Status field shall be omitted. Otherwise, the Remote Device shall implement the following processing.

5053
5054
5055
5056
5057
5058

Upon receipt of and after support for the Mgmt_Bind_req has been verified, the Remote Device shall perform an APSME-GET.request (for the *apsBindingTable* attribute) and process the resulting APSME-GET.confirm (containing the *apsBindingTable* attribute) to create the Mgmt_Bind_rsp command. The Mgmt_Bind_rsp command shall contain the same status that was contained in the APSME-GET.confirm primitive and if this was not SUCCESS, all parameter fields after the status field shall be omitted.

5059 From the *apsBindingTable* attribute, the binding table shall be accessed, starting with the index specified by
5060 StartIndex, and moved to the BindingTableList field of the Mgmt_Bind_rsp command. The entries reported
5061 from the binding table shall be those, starting with StartIndex and including whole BindingTableList rec-
5062 ords (see Table 2.134) until the MSDU size limit, i.e., *aMaxMACFrameSize* (see [B1]), is reached. Within
5063 the Mgmt_Bind_Rsp command, the BindingTableEntries field shall represent the total number of Binding
5064 Table entries in the Remote Device. The BindingTableListCount field shall be the number of entries re-
5065 ported in the BindingTableList field of the Mgmt_Bind_req command.

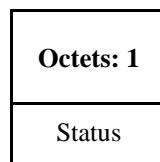
5066 **2.4.4.4.2 Effect on Receipt**

5067 The local device is notified of the results of its attempt to obtain the binding table.

5068 **2.4.4.4.5 Mgmt_Leave_rsp**

5069 The Mgmt_Leave_rsp command (ClusterID=0x8034) shall be formatted as illustrated in Figure 2.100.

5070 **Figure 2.100 Format of the Mgmt_Leave_rsp Command Frame**



5071 Table 2.135 specifies the fields of the Mgmt_Leave_rsp command frame.

5072 **Table 2.135 Fields of the Mgmt_Leave_rsp Command**

Name	Type	Valid Range	Description
Status	Integer	NOT_SUPPORTED, NOT_AUTHORIZED or any status code returned from the NLME-LEAVE.confirm primitive	The status of the Mgmt_Leave_req command.

5073 **2.4.4.4.5.1 When Generated**

5074 The Mgmt_Leave_rsp is generated in response to a Mgmt_Leave_req. Stacks certified prior to revision 21
5075 may or may not support this command. If this management command is not supported, a status of
5076 NOT_SUPPORTED shall be returned. All stacks certified to revision 21 and later must support this com-
5077 mand.

5078 **2.4.4.4.5.2 Effect on Receipt**

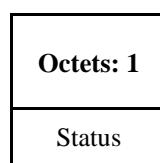
5079 Upon receipt of the Mgmt_Leave_rsp the device may parse the Status field to determine whether or not the
5080 remote device accepted the leave request.

5081

5082 **2.4.4.4.6 Mgmt_Direct_Join_rsp**

5083 The Mgmt_Direct_Join_rsp (ClusterID=0x8035) shall be formatted as illustrated in Figure 2.101.

5084 **Figure 2.101 Format of the Mgmt_Direct_Join_rsp Command Frame**



5085

5086 Table 2.136 specifies the fields of the Mgmt_Direct_Join_rsp command frame.

5087 **Table 2.136 Fields of the Mgmt_Direct_Join_rsp Command**

Name	Type	Valid Range	Description
Status	Integer	NOT_SUPPORTED, NOT_AUTHORIZED or any status code returned from the NLME-DIRECT-JOIN.confirm primitive	The status of the Mgmt_Direct_Join_req command.

5088 **2.4.4.4.6.1 When Generated**

5089 The Mgmt_Direct_Join_rsp is generated in response to a Mgmt_Direct_Join_req. If this management
5090 command is not supported, a status of NOT_SUPPORTED shall be returned. Otherwise, the Remote Device
5091 shall implement the following processing.

5092 Upon receipt and after support for the Mgmt_Direct_Join_req has been verified, the Remote Device shall
5093 execute the NLME-DIRECT-JOIN.request to directly associate the DeviceAddress contained in the
5094 Mgmt_Direct_Join_req to the network. The Mgmt_Direct_Join_rsp shall contain the same status that was
5095 contained in the NLME-DIRECT-JOIN.confirm primitive.

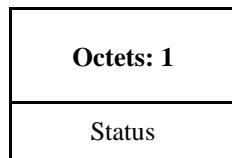
5096 **2.4.4.4.6.2 Effect on Receipt**

5097 Upon receipt and after support for the Mgmt_Direct_Join_req has been verified, the Remote Device shall
5098 execute the NLME-DIRECT-JOIN.request to directly associate the DeviceAddress contained in the
5099 Mgmt_Direct_Join_req to the network.

5100 **2.4.4.4.7 Mgmt_PermitJoining_rsp**

5101 The Mgmt_PermitJoining_rsp command (ClusterID=0x8036) shall be formatted as illustrated in Figure
5102 2.102.

5103 **Figure 2.102 Format of the Mgmt_PermitJoining_rsp Command Frame**



5104

5105 Table 2.137 specifies the fields of the Mgmt_PermitJoining_rsp command frame.

5106

Table 2.137 Fields of the Mgmt_Permit_Joining_rsp Command

Name	Type	Valid Range	Description
Status	Integer	SUCCESS, INVALID_REQUEST, NOT_AUTHORIZED or any status code returned from the NLME-PERMIT-JOINING.confirm primitive	The status of the Mgmt_Permit_Joining_rsp com- mand.

5107

2.4.4.4.7.1 When Generated

5108
5109
5110
5111
5112
5113
5114

The Mgmt_Permit_Joining_rsp is generated in response to a unicast Mgmt_Permit_Joining_req. In the de-
scription which follows, note that no response shall be sent if the Mgmt_Permit_Joining_req was received
as a broadcast to all routers. If this management command is not permitted by the requesting device, a sta-
tus of INVALID_REQUEST shall be returned. Upon receipt and after support for
Mgmt_Permit_Joining_req has been verified, the Remote Device shall execute the
NLME-PERMIT-JOINING.request. The Mgmt_Permit-Joining_rsp shall contain the same status that was
contained in the NLME-PERMIT-JOINING.confirm primitive.

5115

2.4.4.4.7.2 Effect on Receipt

5116

The status of the Mgmt_Permit_Joining_req command is notified to the requestor.

5117

2.4.4.4.8 Mgmt_Cache_rsp

5118

The Mgmt_Cache_rsp command (ClusterID=0x8037) shall be formatted as illustrated in Figure 2.103.

5119

Figure 2.103 Format of the Mgmt_Cache_rsp Command Frame

Octets: 1	1	1	1	Variable
Status	DiscoveryCacheEntries	startIndex	DiscoveryCacheListCount	DiscoveryCacheList

5120

Table 2.138 specifies the fields of the Mgmt_Cache_rsp command frame.

5121

Table 2.138 Fields of the Mgmt_Cache_rsp Command

Name	Type	Valid Range	Description
Status	Integer	SUCCESS or NOT_SUPPORTED	The status of the Mgmt_Cache_rsp command.
DiscoveryCacheEntries	Integer	0x00 - 0xff	DiscoveryCacheEntries.
startIndex	Integer	0x00 - 0xff	startIndex.

Name	Type	Valid Range	Description
DiscoveryCacheListCount	Integer	0x00 - 0xff	The list shall contain the number of elements given by the DiscoveryCacheListCount parameter.
DiscoveryCacheList	Integer	List of DiscoveryCache descriptors	A list of descriptors, one for each of the Discovery cache devices registered, beginning with the StartIndex element and continuing for DiscoveryCacheListCount, of the registered devices in the Primary Discovery Cache. Each entry shall be formatted as illustrated in Table 2.139.

5123

5124

Table 2.139 DiscoveryCacheList Record Format

Name	Size (Bits)	Valid Range	Description
Extended Address	64	An extended 64-bit IEEE Address	64-bit IEEE Address of the cached device.
Network Address	16	Network address	The 16-bit network address of the cached device.

5125 **2.4.4.4.8.1 When Generated**

5126 The Mgmt_Cache_rsp is generated in response to an Mgmt_Cache_req. If this management command is
 5127 not supported, or the Remote Device is not a Primary Cache Device, a status of NOT_SUPPORTED shall
 5128 be returned and all parameter fields after the Status field shall be omitted. Otherwise, the Remote Device
 5129 shall implement the following processing. Upon receipt of the Mgmt_Cache_req and after support for the
 5130 Mgmt_Cache_req has been verified, the Remote Device shall access an internally maintained list of regis-
 5131 tered ZigBee End Devices utilizing the discovery cache on this Primary Discovery Cache device. The en-
 5132 tries reported shall be those, starting with StartIndex and including whole DiscoveryCacheList records (see
 5133 Table 2.142) until the limit on MSDU size, i.e., $aMaxMACFrameSize$ (see [B1]), is reached. Within the
 5134 Mgmt_Cache_rsp command, the DiscoveryCacheListEntries field shall represent the total number of regis-
 5135 tered entries in the Remote Device. The parameter DiscoveryCacheListCount shall be the number of entries
 5136 reported in the DiscoveryCacheList field of the Mgmt_Cache_rsp command.

5137 **2.4.4.4.8.2 Effect on Receipt**

5138 The local device is notified of the results of its attempt to obtain the discovery cache list.

5139 **2.4.4.4.9 Mgmt_NWK_Update_notify**

5140 The Mgmt_NWK_Update_notify command (ClusterID=0x8038) shall be formatted as illustrated in Figure
 5141 2.104.

5142

Figure 2.104 Format of the Mgmt_NWK_Update_notify Command Frame

Oc-tets: 1	4	2	2	1	Variable
Status	ScannedChann-nels	TotalTransmis-sions	TransmissionFail-ures	ScannedChannelsList-Count	Ener-gyValues

5143

5144 Table 2.140 specifies the fields of the Mgmt_NWK_Update_notify command frame.

Table 2.140 Fields of the Mgmt_NWK_Update_notify Command

Name	Type	Valid Range	Description
Status	Integer	SUCCESS, INVALID_REQUEST, NOT_SUPPORTED or any status values returned from the PLME-SET,confirm primitive	The status of the Mgmt_NWK_Update_notify command.
ScannedChannels	Bitmap	0x00000000 - 0xffffffff	List of channels scanned by the request.
TotalTransmissions	Integer	0x0000 -0xffff	Count of the total transmissions reported by the device.
TransmissionFailures	Integer	x0000 -0xffff	Sum of the total transmission failures reported by the device.
ScannedChannelsListCount	Integer	0x00 - 0xff	The list shall contain the number of rec- ords contained in the EnergyValues parameter.
EnergyValues	Integer	List of ED values each of which can be in the range of 0x00 - 0xff	The result of an energy measurement made on this channel in accordance with [B1].

5146

2.4.4.4.9.1 When Generated

5147

5148 The Mgmt_NWK_Update_notify is provided to enable ZigBee devices to report the condition on local
5149 channels to a network manager. The scanned channel list is the report of channels scanned and it is fol-
5150 lowed by a list of records, one for each channel scanned, each record including one byte of the energy level
measured during the scan, or 0xff if there is too much interference on this channel.

5151

5152

When sent in response to a Mgmt_NWK_Update_req command the status field shall represent the status of
the request. When sent unsolicited the status field shall be set to SUCCESS.

5153 **2.4.4.4.9.2 Effect on Receipt**

5154 The local device is notified of the local channel conditions at the transmitting device, or of its attempt to
5155 update network configuration parameters.

5156 **2.4.5 ZDP Enumeration Description**

5157 This section explains the meaning of the enumerations used in the ZDP. Table 2.141 shows a description of
5158 the ZDP enumeration values.

5159 **Table 2.141 ZDP Enumerations Description**

Enumeration	Value	Description
SUCCESS	0x00	The requested operation or transmission was completed successfully.
-	0x01-0x7f	Reserved.
INV_REQUESTTYPE	0x80	The supplied request type was invalid.
DEVICE_NOT_FOUND	0x81	The requested device did not exist on a device following a child descriptor request to a parent.
INVALID_EP	0x82	The supplied endpoint was equal to 0x00 or 0xff.
NOT_ACTIVE	0x83	The requested endpoint is not described by a simple descriptor.
NOT_SUPPORTED	0x84	The requested optional feature is not supported on the target device.
TIMEOUT	0x85	A timeout has occurred with the requested operation.
NO_MATCH	0x86	The end device bind request was unsuccessful due to a failure to match any suitable clusters.
-	0x87	Reserved.
NO_ENTRY	0x88	The unbind request was unsuccessful due to the coordinator or source device not having an entry in its binding table to unbind.
NO_DESCRIPTOR	0x89	A child descriptor was not available following a discovery request to a parent.
INSUFFICIENT_SPACE	0x8a	The device does not have storage space to support the requested operation.

Enumeration	Value	Description
NOT_PERMITTED	0x8b	The device is not in the proper state to support the requested operation.
TABLE_FULL	0x8c	The device does not have table space to support the operation.
NOT_AUTHORIZED	0x8d	The device has rejected the command due to security restrictions.
DEVICE_BINDING_TABLE_FULL	0x8e	The device does not have binding table space to support the operation.
-	0x8f-0xff	Reserved.

5160 2.4.6 Conformance

5161 When conformance to this Profile is claimed, all capabilities indicated mandatory for this Profile shall be
5162 supported in the specified manner (process mandatory). This also applies to optional and conditional capa-
5163 bilities, for which support is indicated, and is subject to verification as part of the ZigBee certification pro-
5164 gram.

5165 2.5 The ZigBee Device Objects (ZDO)

5166 2.5.1 Scope

5167 This section describes the concepts, structures, and primitives needed to implement a ZigBee Device Ob-
5168 jects application on top of a ZigBee Application Support Sub-layer (section 2.2) and ZigBee Network Lay-
5169 er (Chapter 3).

5170 ZigBee Device Objects are applications which employ network and application support layer primitives to
5171 implement ZigBee End Devices, ZigBee Routers, and ZigBee Coordinators.

5172 The ZigBee Device Object Profile employs Clusters to describe its primitives. The ZigBee Device Profile
5173 Clusters do not employ attributes and are analogous to messages in a message transfer protocol. Cluster
5174 identifiers are employed within the ZigBee Device Profile to enumerate the messages employed within
5175 ZigBee Device Objects.

5176 ZigBee Device Objects also employ configuration attributes. The configuration attributes within ZigBee
5177 Device Objects are attributes set by the application or stack profile. The configuration attributes are also not
5178 related to the ZigBee Device Profile, though both the configuration attributes and the ZigBee Device Pro-
5179 file are employed with ZigBee Device Objects.

5180 2.5.2 Device Object Descriptions

5181 The ZigBee Device Objects are an application solution residing within the Application Layer (APL) and
5182 above the Application Support Sub-layer (APS) in the ZigBee stack architecture as illustrated in Figure 1.1.

5183 The ZigBee Device Objects are responsible for the following functions:

- 5184 • Initializing the Application Support Sublayer (APS), Network Layer (NWK), Security Service Provider (SSP) and any other ZigBee device layer other than the end applications residing over Endpoints 1-254.
- 5185 • Assembling configuration information from the end applications to determine and implement the functions described in the following sections.
- 5186
- 5187
- 5188

5189 **2.5.2.1 Primary Discovery Cache Device Operation**

5190 The Primary Discovery Cache device is designated through configuration of the device and advertisement in the Node Descriptor. The Primary Discovery Cache device operates as a state machine with respect to clients wishing to utilize the services of the Primary Discovery Cache. The following states and operations, as described in Figure 2.105, shall be supported by the Primary Discovery Cache device:

5191

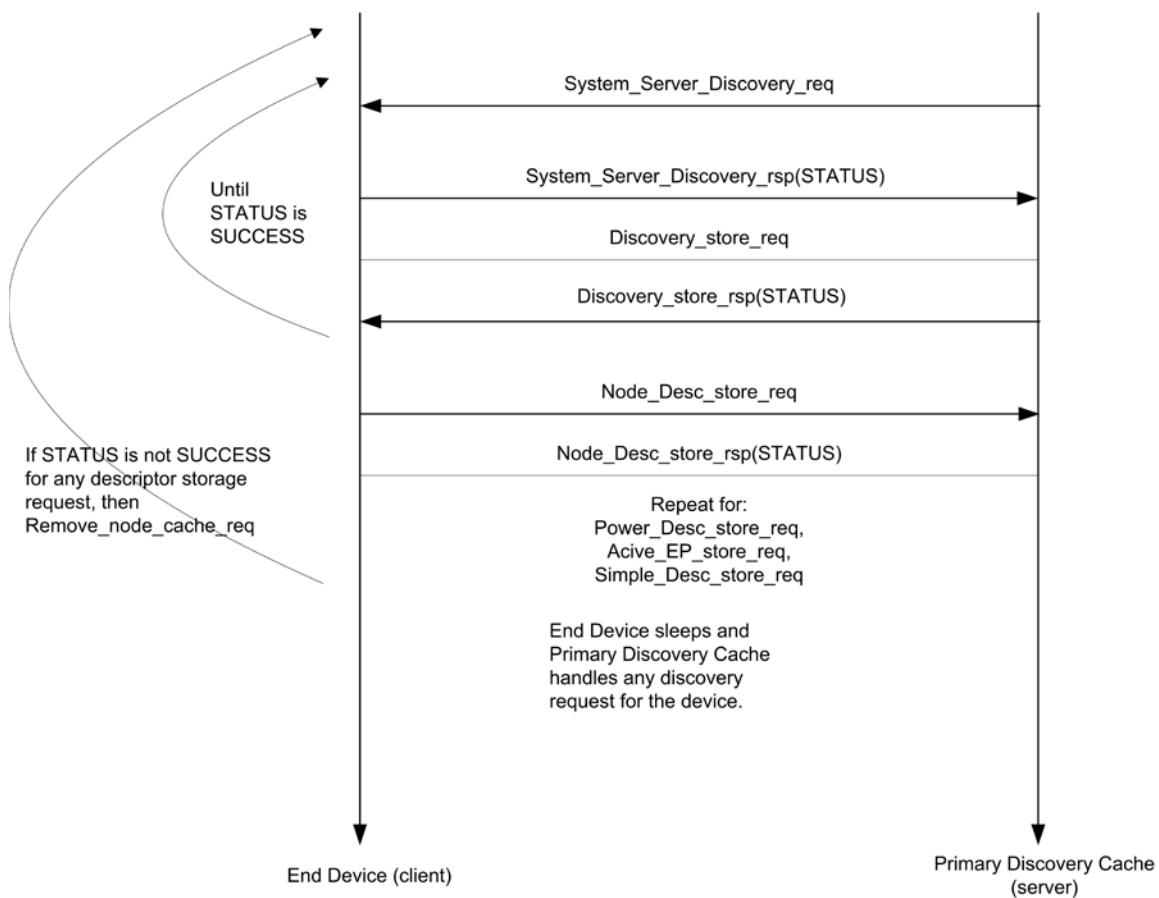
5192

5193

- 5194 • Undiscovered:
- 5195 ◦ The client employs the Find Node Cache request, broadcast to all devices for which macRx-OnWhenIdle=TRUE to determine if there is an existing discovery cache entry for the Local Device. If a discovery cache device responds to the request, the Local Device may update the discovery information and shall transition to the Registered state.
- 5196
- 5197 ◦ The client employs the radius limited message System Server Discovery request, broadcast to all devices for which macRxOnWhenIdle = TRUE, to locate a Primary Discovery Cache device within the radius supplied by the request.
- 5198
- 5199
- 5200
- 5201
- 5202 • Discovered:
- 5203 ◦ The client employs the unicast Discovery store request directed to the Discovery Cache device containing the sizes of the discovery cache information it wishes to store. The Discovery Cache Device will respond with a SUCCESS, INSUFFICIENT_SPACE or NOT_SUPPORTED.
- 5204
- 5205
- 5206 • Registered:
- 5207 ◦ This state is reached when a SUCCESS status was received by the client from the Discovery Cache device from a previous Discovery cache request or the Find Node Cache request found a pre-existing discovery cache entry. The client must now upload its discovery information using the Node Descriptor store request, Power Descriptor store request, Active Endpoint store request, and Simple Descriptor store requests to enable the Primary Discovery Cache device to fully respond on its behalf.
- 5208
- 5209
- 5210
- 5211
- 5212
- 5213 • Unregistered:
- 5214 ◦ The client (or any other device) may request to be unregistered. The Remove Node Cache request removes the device from the Primary Discovery Cache device. The Primary Cache Device responds to device and service discovery requests for all registered clients it supports. The Find Node Cache request is employed by clients wanting to locate the device and service discovery location for a given device of interest. Note that if the discovery information is held by the device itself, that device must also respond to identify itself as the repository of discovery information. See Figure 2.105 for details on state machine processing for the Primary Discovery Cache device.
- 5215
- 5216
- 5217
- 5218
- 5219
- 5220

5221

Figure 2.105 Primary Discovery Cache State Machine



5222

2.5.2.2 Device and Service Discovery

5224 This function shall support device and service discovery within a single PAN. Additionally, for all ZigBee
5225 device types, this function shall perform the following:

- 5226 • Within each network employing sleeping ZigBee End Devices, some ZigBee Routers (or the ZigBee
5227 Coordinator) may be designated as Primary Discovery Cache Devices as described by their Node De-
5228 scriptor. These Primary Cache Devices are themselves discoverable and provide server services to up-
5229 load and store discovery information on behalf of sleeping ZigBee End Devices. Additionally, the
5230 Primary Cache Devices respond to discovery requests on behalf of the sleeping ZigBee End Devices.
5231 Each Primary Discovery Cache Device shall be either a ZigBee Router or the ZigBee Coordinator.
- 5232 • For ZigBee End Devices which intend to sleep as indicated by:Config_Node_Power, Device and Ser-
5233 vice Discovery may manage upload and storage of the NWK Address, IEEE Address, Active End-
5234 points, Simple Descriptors, Node Descriptor, and Power Descriptor onto a Primary Discovery Cache
5235 device selected by the ZigBee End Device to permit device and service discovery operations on these
5236 sleeping devices.
- 5237 • For the ZigBee Coordinator and ZigBee Routers designated as Primary Discovery Cache Devices, this
5238 function shall respond to discovery requests on behalf of sleeping ZigBee End Devices who have reg-
5239 istered and uploaded their discovery information.
- 5240 • For all ZigBee devices, Device and Service Discovery shall support device and service discovery re-
5241 quests from other devices and permit generation of requests from their local Application Objects. Note
5242 that Device and Service Discovery services may be provided by the Primary Discovery Cache devices
5243 on behalf of other ZigBee End Devices. In cases where the Primary Discovery Cache Device is the
5244 target of the request, the NWKAddrOfInterest or Device of Interest fields shall be filled in the request

and/or response to differentiate the target of the request from the device that is the target of discovery.
The following discovery features shall be supported:

- Device Discovery:
 - Based on a unicast inquiry of a ZigBee Coordinator or ZigBee Router's IEEE address, the IEEE Address of the requested device plus, optionally, the NWK Addresses of all associated devices shall be returned.
 - Based on a unicast inquiry of a ZigBee End Device's IEEE address, the IEEE Address of the requested device shall be returned.
 - Based on a broadcast inquiry (of any broadcast address type) of a ZigBee Coordinator or ZigBee Router's NWK Address with a supplied IEEE Address, the NWK Address of the requested device plus, optionally, the NWK Addresses of all associated devices shall be returned.
 - Based on a broadcast inquiry (of any broadcast address type) of a ZigBee End Device's NWK Address with a supplied IEEE Address, the NWK Address of the requested device shall be returned. The responding device shall employ APS acknowledged service for the unicast response to the broadcast inquiry.
- Service Discovery: Based on the following inputs, the corresponding responses shall be supplied:
 - NWK address plus Active Endpoint query type – Specified device shall return the endpoint number of all applications residing in that device. Should the list of active endpoints exceed the ASDU size and where fragmentation is not supported on the server device, an extended version of the query type is also provided to return the full list through multiple requests.
 - NWK address or broadcast address (of any broadcast address type) plus Service Match including Profile ID and, optionally, Input and Output Clusters – Specified device matches Profile ID with all active endpoints to determine a match. If no input or output clusters are specified, the endpoints that match the request are returned. If input and/or output clusters are provided in the request, those are matched as well, and any matches are provided in the response with the list of endpoints on the device providing the match. The responding device shall employ APS acknowledged service for the unicast response to the broadcast inquiry. By convention, in cases where the application profile enumerates input clusters and their response output clusters with the same cluster identifier, the application profile shall list only the input cluster within the Simple Descriptor for the purposes of Service Discovery.
 - NWK address plus Node Descriptor or Power Descriptor query type – Specified device shall return the Node or Power Descriptor for the device.
 - NWK address, Endpoint Number plus Simple Descriptor query type – Specified address shall return the Simple Descriptor associated with that Endpoint for the device. Should the list of input and/or output clusters exceed the ASDU size capacity to return the Simple Descriptor in a single packet an extended version of the query type is also provided to return the full list through multiple requests.
 - Optionally, NWK address plus Complex or User Descriptor query type
 - If supported, specified address shall return the Complex or User Descriptor for the device

2.5.2.3 Security Manager

This function determines whether security is enabled or disabled and, if enabled, shall perform the following:

- Transport Key
- Request Key
- Update Device
- Remove Device
- Switch Key

5293 The Security Manager function addresses the Security Services Specification (Chapter 4). The Security
5294 Management entity, implemented by APSME primitive calls by ZDO, performs the following:

- 5295 • Transports the NWK Key from the Trust Center using secured communication with the Trust Center.
5296 This step employs the APSME-TRANSPORT-KEY primitive.
- 5297 • Establishes or transports Link Keys, as required, with specific devices in the network. These steps em-
5298 ploy the APSME-TRANSPORT-KEY and/or APSME-REQUEST-KEY primitives.
- 5299 • Informs the Trust Center of any devices that join the network using the APSME-UPDATE-DEVICE
5300 primitives. This function is only performed if the device is a ZigBee router.
- 5301 • Permits devices to obtain keys from the Trust Center using the APSME-REQUEST-KEY primitives.
- 5302 • Permits the Trust Center to remove devices from the network using the APSME-REMOVE-DEVICE
5303 primitives.
- 5304 • Permits the Trust Center to switch the active network key using the APSME-SWITCH-KEY primi-
5305 tives.
- 5306 •

5307 **2.5.2.4 Network Manager**

5308 This function shall implement the ZigBee Coordinator, ZigBee Router, or ZigBee End Device logical de-
5309 vice types according to configuration settings established either via a programmed application or during in-
5310 stallation. If the device type is a ZigBee Router or ZigBee End Device, this function shall provide the abil-
5311 ity to select an existing PAN to join and implement procedures which permit the device to rejoin if network
5312 communication is lost. If the device type is a ZigBee Coordinator or ZigBee Router, this function shall
5313 provide the ability to select an unused channel for creation of a new PAN. Note that it is possible to deploy
5314 a network without a device pre-designated as ZigBee Coordinator where the first Full Function Device
5315 (FFD) activated assumes the role of ZigBee Coordinator. The following description covers processing ad-
5316 dressed by Network Management:

- 5317 • Permits specification of a channel list for network scan procedures. Default is to specify use of all
5318 channels in the selected band of operation.
- 5319 • Manages network scan procedures to determine neighboring networks and the identity of their ZigBee
5320 coordinators and routers.
- 5321 • Permits selection of a channel to start a PAN (ZigBee Coordinator) or selection of an existing PAN to
5322 join (ZigBee Router or ZigBee End Device).
- 5323 • Supports orphaning and extended procedures to rejoin the network, including support for intra_PAN
5324 portability.
- 5325 • May support direct join. For ZigBee Coordinators and ZigBee Routers, a local version of direct join
5326 may be supported to enable the device to join via the orphaning or rejoin procedures.
- 5327 • May support Management Entities that permit external network management.
- 5328 • Detects and reports interference to support changing network channels.
- 5329 • Manages network interference reporting and selection of a new channel for network operation if inter-
5330 ference exists on the initial channel if the particular node is identified as the network manager for the
5331 overall PAN.

5332 **2.5.2.5 Binding Manager**

5333 The Binding Manager performs the following:

- 5334 • Establishes resource size for the Binding Table. The size of this resource is determined via a pro-
5335 grammmed application or via a configuration attribute defined during installation.
- 5336 • Processes bind requests for adding or deleting entries from the APS binding table.

- 5337 • Supports Bind and Unbind commands from external applications such as those that may be hosted on a
5338 commissioning or network management tool to support assisted binding. Bind and Unbind commands
5339 shall be supported via the ZigBee Device Profile (see clause 2.4).
5340 • For the ZigBee Coordinator, supports the End Device Bind that permits binding on the basis of button
5341 presses or other manual means.
5342 • Permits source devices to register with a primary binding table cache their ability to hold their own
5343 binding table.
5344 • Permits configuration tools to exchange one device for another in all the binding table entries which
5345 refer to it.
5346 • Permits the primary binding table cache to backup and recover individual bind entries or the entire
5347 binding table or the table of source devices holding their own binding tables.

5348 **2.5.2.6 Node Manager**

5349 For ZigBee Coordinators and ZigBee Routers, the Node Management function performs the following:

- 5350 • Permits remote management commands to perform network discovery.
5351 • Provides remote management commands to retrieve the routing table.
5352 • Provides remote management commands to retrieve the binding table.
5353 • Provides a remote management command to have the device leave the network or to direct that another
5354 device leave the network.
5355 • Provides a remote management command to retrieve the LQI for neighbors of the remote device.
5356 • Provides a remote management command to Permit or disallow joining on particular routers or to gen-
5357 erally allow or disallow joining via the Trust Center.

5358 **2.5.2.7 Group Manager**

5359 The Group Manager performs the following:

- 5360 • Provides for inclusion of application objects within the local device into groups under application con-
5361 trol.
5362 • Provides for removal of application objects within the local device from group membership under ap-
5363 plication control.

5364 **2.5.3 Layer Interface Description**

5365 Unlike other device descriptors for applications residing above Endpoints 1-254, the ZigBee Device Ob-
5366 jects (ZDO) interface to the APS via the APSME-SAP in addition to the APSDE-SAP. ZDO communicates
5367 over Endpoint 0 using the APSDE-SAP via Profiles like all other applications. The Profile used by ZDO is
5368 the ZigBee Device Profile (see clause 2.4). ZDO frames shall not be fragmented.

5369 ZigBee Device Objects shall employ Endpoint 0 as the source and destination endpoint in any transmitted
5370 ZigBee Device Profile request frames, and shall expect Endpoint 0 as the source and destination endpoint
5371 in any received response frames.

5372 **2.5.4 System Usage**

5375

5376 **2.5.4.1 Object Overview**

5377 ZigBee Device Objects contain six Objects:

- 5378 • Device and Service Discovery
5379 • Network Manager
5380 • Binding Manager
5381 • Security Manager
5382 • Node Manager
5383 • Group Manager

5384 Table 2.142 describes these ZigBee Device Objects.

5385 **Table 2.142 ZigBee Device Objects**

Object		Description
Name	Status	
:Device_and_Service_Discovery	M	Handles device and service discovery.
:Network_Manager	M	Handles network activities such as network discovery, leaving/joining a network, resetting a network connection and creating a network.
:Binding_Manager	O	Handles end device binding, binding and unbinding activities.
:Security_Manager	O	Handles security services such as key loading, key establishment, key transport and authentication.
:Node_Manager	O	Handles management functions.
:Group Manager	O	Handles management of groups

5386 **2.5.4.2 Optional and Mandatory Objects and Attributes**

5387 Objects listed as Mandatory shall be present on all ZigBee devices. However, for certain ZigBee logical
5388 types, Objects listed as Optional for all ZigBee devices may be Mandatory in specific logical device types.
5389 For example, the NLME-NETWORK-FORMATION.request within the Network_Manager object is in a
5390 Mandatory object and is an Optional attribute, though the attribute is required for ZigBee Coordinator logical
5391 device types. The introduction section of each Device Object section will detail the support requirements
5392 for Objects and Attributes by logical device type.

5393 **2.5.4.3 Security Key Usage**

5394 ZigBee Device Objects may employ security for packets created by ZigBee Device Profile primitives.
5395 These application packets using APSDE on Endpoint 0 shall utilize the APSDE Security Service Provider
5396 interface like all other Application Objects.

5397 **2.5.4.4 Public and Private Methods**

5398 Methods that are accessible to any endpoint application on the device are called public methods. Private
5399 methods are only accessible to the Device Application on endpoint 0 and not to the end applications (which
5400 run on endpoints 1 through 254).

5401 **2.5.4.5 State Machine Functional Descriptions**

5402 **2.5.4.5.1 ZigBee Coordinator**

5403 **2.5.4.5.1.1 Initialization**

5404 The implementation shall set the startup-related IB attributes shown in Table 2.143 to values that reflect the
5405 desired startup behavior for the device. In particular, the *apsDesignatedCoordinator* attribute of the IB shall
5406 be set to TRUE. If the device implements more than one option for ZigBee protocol version or stack pro-
5407 file, it shall choose a single value for each and set *nwkProtocolVersion* and *nwkStackProfile* accordingly.
5408 Additionally, provision shall be made to provide configuration elements to describe the Node Descriptor,
5409 Power Descriptor, Simple Descriptor for each active endpoint and application plus the list of active end-
5410 points. These configurations shall be embodied in :Config_Node_Descriptor, :Config_Power_Descriptor,
5411 and
5412 :Config_Simple_Descriptors. If the :Config_Node_Descriptor configuration object indicates that this de-
5413 vice is a Primary Discovery Cache device, the device shall be configured to process server commands for
5414 the ZigBee Device Profile associated with requests to the Primary Discovery Cache and shall operate ac-
5415 cording to the state machine description provided in section 2.5.2.1.

5416 If supported, provision shall be made to supply configuration elements for the Complex Descriptor, User
5417 Descriptor, and the maximum number of bind entries. These elements shall be embodied in
5418 :Config_Complex_Descriptor, :Config_User_Descriptor, and :Config_Max_Bind.

5419 To start as a ZigBee coordinator, the device application shall execute the startup procedure described in
5420 section 2.5.4.5.6.2 with startup attributes set as described above. This should have the effect of executing
5421 the procedure for network formation described in section 3.6.1.1. The device application shall set the
5422 *nwkSecurityLevel* and *nwkAllFresh* NIB attributes according to the values established by convention within
5423 the Stack Profile employed by the device. The device application shall check the return status via the
5424 NLME-NETWORK-FORMATION.confirm to verify successful creation of the PAN. The
5425 :Config_Permit_Join_Duration shall be set according to the default attribute value supplied using the
5426 NLME-PERMIT-JOINING.request. Additionally, the *nwkNetworkBroadcastDeliveryTime* and *nwk-*
5427 *TransactionPersistenceTime* Network Information Block attributes (see section 3.6.2) shall be set with :
5428 *Config_NWK_BroadcastDeliveryTime* and *Config_NWK_TransactionPersistenceTime* respectively (see
5429 section 2.5.5).

5430 Provision shall be made to ensure APS primitive calls from the end applications over EP 1 through EP 254
5431 return appropriate error status values prior to completion of the Initialization state by ZigBee Device Ob-
5432 jects and transition to the normal operating state.

5433 **2.5.4.5.1.2 Normal Operating State**

5434 In this state, the ZigBee Coordinator shall process the list of direct joined addresses in
5435 :Config_NWK_Join_Direct_Addrs by issuing an NLME-DIRECT-JOIN.request for each included address
5436 in the list. Processing of the direct joined addresses shall employ the :Config_Max_Assoc attribute in eval-
5437 uating whether to successfully process a direct joined address within :Config_NWK_Join_Direct_Addrs.

5438 The ZigBee coordinator shall allow other devices to join the network based on the configuration items
5439 :Config_Permit_Join_Duration and :Config_Max_Assoc. When a new device joins the network, the de-
5440 vice application shall be informed via the NLME-JOIN.indication. Should the device be admitted to the
5441 PAN, the ZigBee coordinator shall indicate this via the NLME-JOIN.confirm with SUCCESS status.

5442 The ZigBee coordinator shall respond to any device discovery or service discovery operations requested of
5443 its own device, and if it is designated as a Primary Discovery Cache device, shall also respond on behalf of
5444 registered devices that have stored discovery information. The device application shall also ensure that the
5445 number of binding entries does not exceed the :Config_Max_Bind attribute.

5446 The ZigBee coordinator shall support the NLME-PERMIT-JOINING.request and
5447 NLME-PERMIT-JOINING.confirm to permit application control of network join processing.

5448 The ZigBee coordinator shall support the NLME-LEAVE.request and NLME-LEAVE.indication employ-
5449 ing the :Config_NWK_Leave_removeChildren attribute where appropriate to permit removal of associated
5450 devices under application control. Conditions that lead to removal of associated devices may include lack
5451 of security credentials, removal of the device via a privileged application or detection of exception.

5452 The ZigBee coordinator shall maintain a list of currently associated devices and facilitate support of orphan
5453 scan and rejoin processing to enable previously associated devices to rejoin the network. The ZigBee coordi-
5454 nator may support the ability for devices to be directly included in the network via the
5455 NLME-DIRECT-JOIN.request and NLME-DIRECT-JOIN.confirm. This feature shall permit lists of
5456 ZigBee IEEE addresses to be provided to the ZigBee coordinator and for those addresses to be included as
5457 previously associated devices. It shall be possible for ZigBee devices with those addresses to directly join
5458 the network via orphaning or rejoin procedures rather than associating directly.

5459 The ZigBee coordinator shall support the NLME-NWK-STATUS.indication and process those notifications
5460 per clause 3.2.2.30.

5461 The ZigBee coordinator shall process End_Device_Bind_req from ZigBee Routers and ZigBee End Devic-
5462 es. Upon receipt of an End_Device_Bind_req, the ZigBee Coordinator shall use the
5463 :Config_EndDev_Bind_Timeout value in the attribute and await a second End_Device_Bind_req. Should
5464 the second indication arrive within the timeout period, the ZigBee coordinator shall match the Profile ID in
5465 the two indications (see section 2.3.3.2). If the Profile IDs in the two indications do not match, an appropri-
5466 ate error status is returned to each device via End_Device_Bind_rsp. Should the Profile IDs match, the
5467 ZigBee Coordinator shall match the AppInClusterLists and AppOutClusterLists in the two indications.
5468 Cluster IDs in the AppInClusterList of the first indication which match Cluster IDs in the AppOutCluster-
5469 List of the second indication shall be saved in a list for inclusion in the resulting Bind_req notifying the de-
5470 vices of the match.

5471 The ZigBee coordinator shall process Device_annce messages from other ZigBee devices. Upon receipt of
5472 a Device_annce where *nwkUseTreeRouting* is TRUE, the ZigBee coordinator shall check all internal tables
5473 holding 64-bit IEEE addresses for devices within the PAN for a match with the address supplied in the De-
5474 vice_annce message. If a match is detected, the ZigBee coordinator shall update its *nwkAddressMap* attrib-
5475 ute of the NIB corresponding to the matched 64-bit IEEE address to reflect the updated 16-bit NWK ad-
5476 dress contained in the Device_annce. Upon receipt of a Device_annce where *nwkUseTreeRouting* is
5477 FALSE, the ZigBee Coordinator shall employ the address conflict resolution procedure detailed in sec-
5478 tion 3.6.9.

5479 The ZigBee coordinator may generate APSME-AUTHENTICATE.requests under application control from
5480 other application objects, and may process and respond to APSME-AUTHENTICATE.indications from
5481 other devices. The ZigBee coordinator shall supply APSME-AUTHENTICATE.confirms to application
5482 objects whose requests have been processed.

5483 **2.5.4.5.1.3 Trust Center Operation**

5484 The network device pointed to by the address in *apsTrustCenterAddress* shall function as the Trust Center
5485 when security is enabled on the network.

5486 The Trust Center operation is defined within section 4.6.2.

5487 **2.5.4.5.2 ZigBee Router**

5488 **2.5.4.5.2.1 Initialization**

5489 The implementation shall set the startup-related IB attributes shown in Table 2.143 to values that reflect the
5490 desired startup behavior for the device. In particular, the *apsDesignatedCoordinator* attribute of the IB shall
5491 be set to FALSE. If the :Config_Node_Descriptor configuration object indicates that this device is a Primary
5492 Discovery Cache device, the device shall be configured to process server commands for the ZigBee
5493 Device Profile associated with requests to the Primary Discovery Cache and shall operate according to the
5494 state machine description provided in section 2.5.2.1.

5495 If supported, provision shall be made to supply configuration elements for the Complex Descriptor, User
5496 Descriptor, and the maximum number of bind entries.. These elements shall be embodied in
5497 :Config_Complex_Descriptor, :Config_User_Descriptor, and :Config_Max_Bind.

5498 To start as a ZigBee router, the device application shall execute the startup procedure described in section
5499 2.5.4.5.6.2 with startup attributes set as described above. This should have the effect of executing either the
5500 procedure for network rejoin described in section 3.6.1.4.2 or else the full procedure for network join
5501 through MAC association described in section 3.6.1.4.1. The NLME-NETWORK-DISCOVERY.request
5502 procedure shall be implemented :Config_NWK_Scan_Attempts, each separated in time by
5503 :Config_NWK_Time_btwn_Scans. The purpose of repeating the
5504 NLME-NETWORK-DISCOVERY.request is to provide a more accurate neighbor list and associated link
5505 quality indications to the NWK layer. Specification of the algorithm for selection of the PAN shall be left
5506 to the profile description and may include use of the Extended PAN ID, operational mode of the network,
5507 identity of the ZigBee Router or Coordinator identified on the PAN, depth of the ZigBee Router on the
5508 PAN from the ZigBee Coordinator for the PAN, capacity of the ZigBee Router or Coordinator, the routing
5509 cost, or the Protocol Version Number (these parameters are supplied by the
5510 NLME-NETWORK-DISCOVERY.confirm and the beacon payload).

5511 The ZigBee router may join networks employing the current protocol version number or may join networks
5512 employing a previous protocol version number, under application control, if backward compatibility is
5513 supported in the device. A single ZigBee PAN shall consist of devices employing only a single protocol
5514 version number (networks with devices employing different protocol version numbers and frame formats
5515 within the same PAN are not permitted). An optional configuration attribute,
5516 :Config_NWK_alt_protocol_version, provides the protocol version numbers which the device may choose
5517 to employ other than the current protocol version number. Once the ZigBee router chooses a PAN and a
5518 specific protocol version number, it shall employ that protocol version number as its *nwkProtocolVersion*.
5519 Additionally, the ZigBee router shall then adhere to all frame formats and processing rules supplied by the
5520 version of the ZigBee Specification employing that protocol version number.

5521 The :Config_Permit_Join_Duration shall be set according to the default parameter value supplied using
5522 NLME-PERMIT-JOINING.request. The router shall support the NLME-START-ROUTER.request and
5523 NLME-START-ROUTER.confirm to begin operations as a router within the PAN it has joined. Additionally,
5524 the *nwkNetworkBroadcastDeliveryTime* and *nwkTransactionPersistenceTime* Network Information
5525 Block attributes (see section 3.6.2) shall be set with :Config_NWK_BroadcastDeliveryTime and
5526 :Config_NWK_TransactionPersistenceTime respectively (see section 2.5.5).

5527 Provision shall be made to ensure APS primitive calls from the end applications over EP 1 through EP 254
5528 return appropriate error status values prior to completion of the Initialization state by ZigBee Device
5529 Objects and transition to the normal operating state.

5530 If the network has security enabled, the device shall wait for successful acquisition of the NWK key to start
5531 functioning as a router in the network. See section 4.6.2 for details on Trust Center operations.

5532 The device application shall set the *nwkSecurityLevel* NIB attribute to the values used in the network and
5533 begin functioning as a router using NLME-START-ROUTER.req.

5534 **2.5.4.5.2.2 Normal Operating State**

5535 In this state, the ZigBee router shall allow other devices to join the network based on the configuration
5536 items :Config_Permit_Join_Duration and :Config_Max_Assoc. When a new device joins the network, the
5537 device application shall be informed via the NLME-JOIN.indication attribute. Should the device be admitted
5538 to the PAN, the ZigBee router shall indicate this via the NLME-JOIN.confirm with SUCCESS status. If
5539 security is enabled on the network, the device application shall inform the Trust Center via the
5540 APSME-UPDATE-DEVICE.request.

5541 Orphan indications for which this device is not the parent are notified to the ZDO from the NWK layer by
5542 receipt of an NLME-JOIN.indication primitive with parameter IsParent set to value FALSE. The mechanism
5543 by which this is handled is described in section 2.5.4.5.4.

5544 The ZigBee router shall respond to any device discovery or service discovery operations requested of its
5545 own device, and if it is designated as a Primary Discovery Cache device, shall also respond on behalf of
5546 registered devices that have stored discovery information. The device application shall also ensure that the
5547 number of binding entries does not exceed the :Config_Max_Bind attribute.

5548 ZigBee router shall request the Trust Center to update its NWK key via the
5549 APSME-REQUEST-KEY.request. The ZigBee router shall support
5550 APSME-TRANSPORT-KEY.indication to receive keys from the Trust Center.

5551 The ZigBee router shall support the NLME-PERMIT-JOINING.request and
5552 NLME-PERMIT-JOINING.confirm to permit application control of network join processing.

5553 The ZigBee router shall support the NLME-NWK-STATUS.indication and process those notifications per
5554 section 3.2.2.30.

5555 The ZigBee router shall support the NLME-LEAVE.request and NLME-LEAVE.confirm employing the
5556 :Config_NWK_Leave_removeChildren attribute where appropriate to permit removal of associated devices
5557 under application control. Conditions that lead to removal of associated devices may include lack of security
5558 credentials, removal of the device via a privileged application or detection of exception.

5559 The ZigBee router shall process Device_annce messages from other ZigBee devices. Upon receipt of a
5560 Device_annce where *nwkUseTreeRouting* is TRUE, the ZigBee router shall check all internal tables holding
5561 64-bit IEEE addresses for devices within the PAN for a match with the address supplied in the
5562 Device_annce message. If a match is detected, the ZigBee router shall update its *nwkAddressMap* of the NIB
5563 corresponding to the matched 64-bit IEEE address to reflect the updated 16-bit NWK address contained in the
5564 Device_annce. Upon receipt of a Device_annce where *nwkUseTreeRouting* is FALSE, the ZigBee Router
5565 shall employ the address conflict resolution procedure detailed in section 3.6.9.

5567 The ZigBee router shall maintain a list of currently associated end devices and facilitate support of orphan
5568 scan and rejoin processing to enable previously associated end devices to rejoin the network.

5569 The ZigBee router may decide it has lost contact with the network it was joined to. In this situation, the
5570 router should conduct an active scan to find the network. If the network is found more than once the router
5571 should attempt to rejoin where there is a more recent value of *nwkUpdateId* in the beacon payload.

5572 **2.5.4.5.3 Binding Table Cache Operation**

5573 Any router (including the coordinator) may be designated as either a primary binding table cache or a
5574 backup binding table cache.

5575 It shall respond to the System_Server_Discovery_req primitive to enable other devices to discover it and
5576 use its facilities.

5577 A primary binding table cache shall maintain a binding table and a table of devices registered to cache their
5578 binding tables.

5579 A primary binding table cache shall respond to the Bind_Register_req and Replace_Device_req primitives
5580 described in clause 2.4.3.2.

5581 If a backup binding table cache is available, a primary binding table cache shall use the additional bind
5582 management primitives to backup and restore its binding table and its table of source binding devices.

5583 A backup binding table cache shall maintain a backup of the binding table and table of registered binding
5584 devices for one or more primary binding table caches. It shall support the bind management primitives for
5585 backup and restore of these tables.

5586 **2.5.4.5.4 Operations to Support Intra-PAN Portability**

5587 **2.5.4.5.4.1 Overview**

5588 The operations described in this section are carried out by ZigBee Coordinator and ZigBee Router Devices
5589 for support of intra-PAN portability.

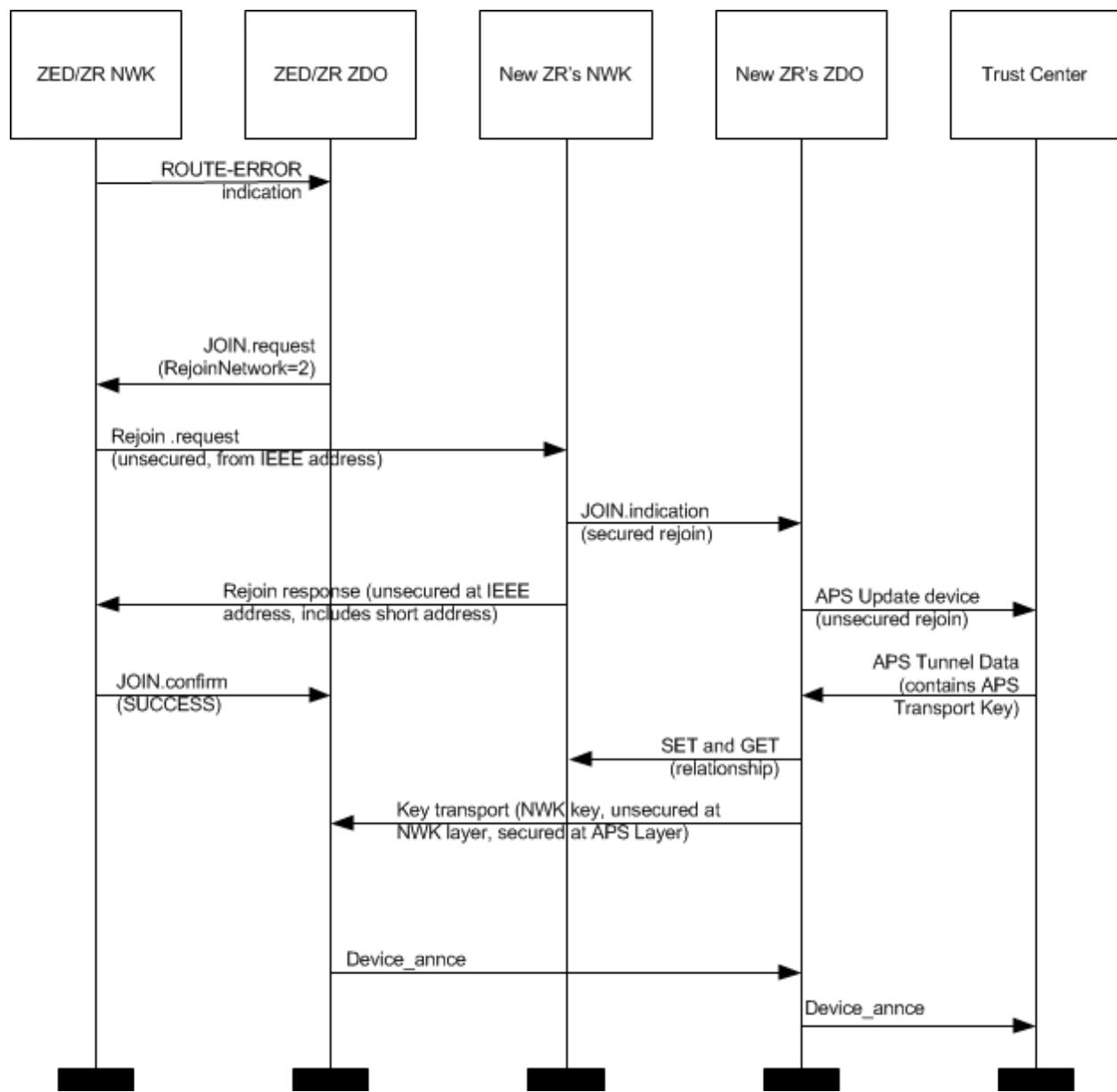
5590 The main steps are summarized as follows:

- 5591 • Detect the problem - The ZDO of the moved device is notified of acknowledgement failures via the
5592 NLME-NWK-STATUS.indication primitive, and identifies a problem.
- 5593 • Carry out the NWK layer rejoin procedure - The ZDO of a moved ZED initiates this process using the
5594 NLME-JOIN.request primitive, either through a secured or un-secured rejoining procedure. The NWK
5595 rejoin procedures closely mirror the MAC association procedure. Note that ZigBee Routers shall also
5596 carry out this procedure periodically if they find that they are no longer in contact with the Trust Cen-
5597 ter.
- 5598 • Security verification - Secured and unsecured protocol steps are described to ensure that the orphaned
5599 device should really be accepted.
- 5600 • Inform the rest of the network - when a device changes parents the steps to complete address conflict
5601 detection in section 3.6.1.9 must be completed. These actions also serve to notify the old parent that
5602 the End Device has changed parents.
- 5603 • Provide a means for parents that were temporarily unavailable and caused the end-device to rejoin are
5604 able to update their child tables once they are back online.

5605 These steps are described in detail in the subsections below. The mechanism is illustrated for secured rejoin
5606 of a ZED in Figure 2.106, trust center rejoin of a ZED in Figure 2.107, and trust center rejoin of a ZR in
Error! Reference source not found. respectively. Note that the NWK and SEC sections on secured and
5607 trust center rejoin (sections 3.2.2.11, 3.2.2.12, 3.2.2.13, 3.6.1.4 and 4.6.3) shall be the authoritative text for
5608 these procedures. The diagrams in this section are provided for illustrative purposes only.

5610

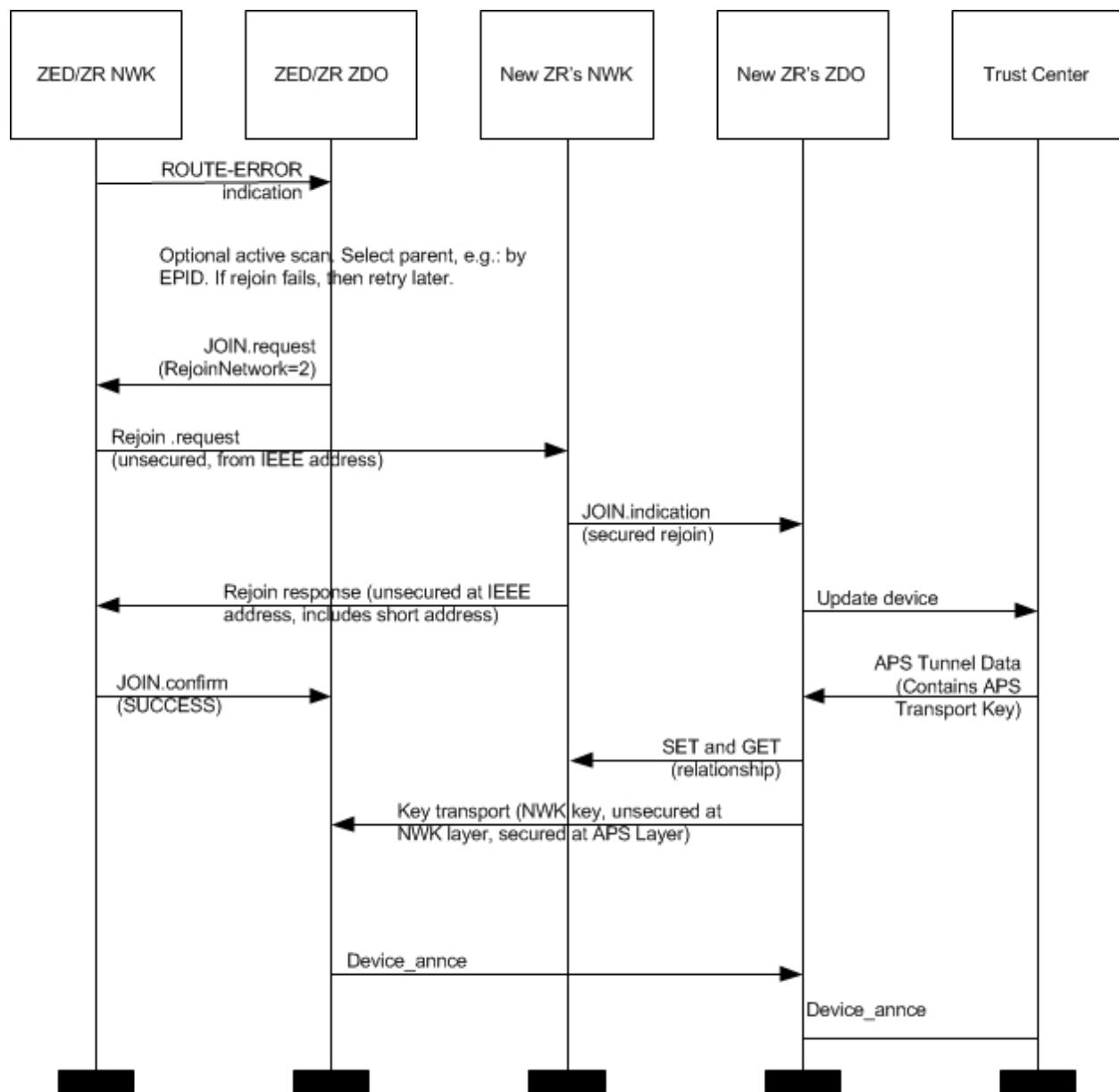
Figure 2.106 Portability Message Sequence Chart: ZED Secured Rejoin



5611
5612

5613

Figure 2.107 Portability Message Sequence Chart: ZR/ZED Trust Center Rejoin



5614

5615

5616

5617

5618 2.5.4.5.4.2 Description of Operations for Security Verification

5619 As for MAC association, a ZigBee Coordinator or ZigBee Router device is informed of a rejoined device
5620 when the NLME issues an NLME-JOIN.indication primitive. This shall be handled in the same way as for
5621 an association indication, except that for a secured rejoin the update device and key transport step.

5622 Full network operation shall not be permitted until the verification steps described below have been carried
5623 out.

5624 Measures shall be taken by a newly (re-)joined node and by its new parent to verify that it is really allowed
5625 to be on this network. Two cases are envisioned:

5626 One or the other is not implemented according to this specification, and should not have joined. The
5627 measures described here allow both sides to revoke the join in this case.

5628 One or the other device is a compromised/hacked device. In the case that security is enabled, the measures
5629 in section 4.6.3.6 are additionally applied so that an unauthorized join is revoked.

5630 This verification is carried out using existing commands. Section 2.5.4.5.4.3 below describes the transmis-
5631 sion of a Device_annce command to the new parent. The new parent shall check that this or some other
5632 message is correctly formed and contains the addressing fields corresponding to the orphaned device. If
5633 security is enabled, then this command shall be secured with the network key, and the new parent shall ver-
5634 ify that all security processing is carried out correctly. If all these checks succeed then the orphaned device
5635 shall become joined to the network. Otherwise, it shall not become joined to the network at this time. As
5636 normal, messages sent from a device not joined to the network shall not be forwarded across the network,
5637 and commands shall not be carried out. Accordingly, the orphaned device shall only become joined to the
5638 network once it receives at least one correctly formed ZigBee message from the new parent. If security is
5639 enabled, this message must be secured with the network key and all security processing must be carried out
5640 correctly. If messages cannot be exchanged in protocol, then the orphaned device shall not become joined
5641 to the network at this time.

5642 **2.5.4.5.4.3 Description of Operations for Informing the Rest of the Network**

5643 If the ZigBee End Device rejoins a new parent using the orphaning of rejoin process it shall complete the
5644 address conflict process in section 3.6.1.9. Upon receiving the Device_annce, all devices shall check their
5645 internal tables holding 64-bit IEEE addresses for devices within the PAN for a match with the address sup-
5646 plied in the Device_annce message. If a match is detected, the device shall update the *nwkAddressMap* at-
5647 tribute of the NIB corresponding to the matched 64-bit IEEE address to reflect the updated 16-bit NWK
5648 address contained in the Device_annce. All devices shall use the NLME-SET and NLME-GET primitives
5649 to update the *nwkNeighborTable* in the NWK NIB. The previous parent of this ZED shall remove the ZED
5650 as one of its children by changing the Relationship field of the *nwkNeighborTable* to 0x04, “previous
5651 child.” Note that any unicast message sent to an address with this status shall result in an
5652 NLME-NWK-STATUS.indication primitive with status code of “Target Device Unavailable”, (see sec-
5653 tion 3.2.2.30). If

5654 *nwkUseTreeRouting* is TRUE, address conflict detection is not provided and parent devices are not per-
5655 mitted, following intra-PAN portability, to remove devices or any other operation that reissue a short address
5656 for use by a child with a different IEEE address. Alternatively, if *nwkUseTreeRouting* is FALSE, address
5657 conflict detection is provided, however, devices will generally keep their existing NWK addresses during
5658 the intra-PAN portability procedure. Also, if the NWK address has changed during the intra-PAN portabil-
5659 ity procedure, the ZDO shall arrange that any IEEE address to short address mappings which have become
5660 known to applications running on this device be updated. This behavior is mandatory, but the mechanism
5661 by which it is achieved is outside the scope of this specification.

5662 **2.5.4.5.5 ZigBee End Device**

5663 **2.5.4.5.5.1 Initialization**

5664 The implementation shall set the startup-related IB attributes shown in Table 2.143 to values that reflect the
5665 desired startup behavior for the device. In particular, the *apsDesignatedCoordinator* attribute of the IB shall
5666 be set to FALSE.

5667 If supported, provision shall be made to supply configuration elements for the Complex Descriptor, User
5668 Descriptor, and the maximum number of bind entries,. These elements shall be embodied in
5669 :Config_Complex_Descriptor, :Config_User_Descriptor, and :Config_Max_Bind. If the device application
5670 set the NLME-JOIN RxOnWhenIdle parameter to FALSE, the end device shall utilize the procedure de-
5671 scribed in section 2.5.2.1 to discover a Primary Discovery Cache device, register with it, and to success-
5672 fully upload its device and service discovery information. To facilitate the process of uploading discovery in-
5673 formation to the Primary Discovery Cache device, the local device may temporarily increase its polling rate
5674 with its parent. Prior to registering with any Primary Discovery Cache device, the end device shall utilize
5675 the Find Node Cache request to ensure it has not previously registered with any other Primary Discovery
5676 Cache device. If a server response indicates the end device has a previous registration, the end device shall
5677 update its discovery cache information on that Primary Discovery Cache device or shall remove its discov-
5678 ery cache information from that previous registration and create a new registration.

5679 To start as a ZigBee end device, the device application shall execute the startup procedure described in sec-
5680 tion 2.5.4.5.6.2 with startup parameters set as described above. This should have the effect of executing ei-
5681 ther the procedure for network rejoin described in section 3.6.1.4.2 or else the full procedure for network
5682 join through MAC association described in section 3.6.1.4.1. The
5683 NLME-NETWORK-DISCOVERY.request procedure shall be implemented
5684 :Config_NWK_Scan_Attempts, each separated in time by
5685 :Config_NWK_Time_btwn_Scans. The purpose of repeating the
5686 NLME-NETWORK-DISCOVERY.request is to provide a more accurate neighbor list and associated link
5687 quality indications to the NWK layer. Specification of the algorithm for selection of the PAN shall be left
5688 to the profile description and may include use of the Extended PAN ID, operational mode of the network,
5689 identity of the ZigBee Router or Coordinator identified on the PAN, depth of the ZigBee Router on the
5690 PAN from the ZigBee Coordinator for the PAN, capacity of the ZigBee Router or Coordinator, the routing
5691 cost, or the Protocol Version Number (these parameters are supplied by the
5692 NLME-NETWORK-DISCOVERY.confirm and the beacon payload).

5693 The ZigBee end device may join networks employing the current protocol version number or may join
5694 networks employing a previous protocol version number, under application control, if backward compati-
5695 bility is supported in the device. A single ZigBee PAN shall consist of devices employing only a single
5696 protocol version number (networks with devices employing different protocol version numbers and frame
5697 formats within the same PAN are not permitted). An optional configuration attribute,
5698 :Config_NWK_alt_protocol_version, provides the protocol version numbers which the device may choose
5699 to employ other than the current protocol version number. Once the ZigBee end device chooses a PAN and
5700 a specific protocol version number, it shall employ that protocol version number as its *nwkProtocolVersion*. Additionally, the ZigBee end device shall then adhere to all frame formats and processing rules
5701 supplied by the version of the ZigBee Specification employing that protocol version number.
5702

5703 If the device application sets the NLME-JOIN RxOnWhenIdle parameter to FALSE, the :Config_NWK_
5704 indirectPollRate shall be used to determine the polling rate for indirect message requests. The
5705 :Config_NWK_indirectPollRate shall be set according to the value established by the application profile(s)
5706 supported on the device. Once polling for indirect message requests is initiated, if communications failure
5707 with the parent is detected determined by failure of indirect message requests
5708 :Config_Parent_Link_Threshold_Retry consecutive attempts, the device application shall employ the net-
5709 work rejoin procedure.

5710 Once the End Device has successfully joined a network, the device shall issue a Device_annce providing its
5711 64-bit IEEE address and 16-bit NWK address.

5712 Provision shall be made to ensure APS primitive calls from the end applications over EP 1 through EP 254
5713 return appropriate error status values prior to completion of the Initialization state by ZigBee Device Ob-
5714 jects and transition to the normal operating state.

5715 If network has security enabled, the device shall wait successful acquisition of the NWK key to start func-
5716 tioning as an end device in the network. See section 4.6.2 for details on Trust Center operations.

5717 **2.5.4.5.5.2 Normal Operating State**

5718 If the device application set the NLME-JOIN RxOnWhenIdle parameter to FALSE, the :Config_NWK_
5719 indirectPollRate shall be used to poll the parent for indirect transmissions while in the normal operating
5720 state. While a fragmented message is being received, the device may temporarily increase its polling rate,
5721 and shall ensure that it polls its parent at least once every macTransactionPersistenceTime seconds.

5722 The ZigBee end device shall respond to any device discovery or service discovery operations requested of
5723 its own device using the attributes described in section 2.5.4.

5724

5725 ZigBee end device shall request the Trust Center to update its NWK key via the
5726 APSME-REQUEST-KEY.request. The ZigBee end device shall support
5727 APSME-TRANSPORT-KEY.indication to receive keys from the Trust Center.

5728 The ZigBee End Device shall process Device_ponce messages from other ZigBee devices. Upon receipt of
5729 a Device_ponce where *nwkUseTreeRouting* is TRUE, the ZigBee End Device shall check all internal tables
5730 holding 64-bit IEEE addresses for devices within the PAN for a match with the address supplied in the
5731 Device_ponce message. If a match is detected, the ZigBee End Device shall update the *nwkAddressMap* of
5732 the NIB corresponding to the matched 64-bit IEEE address to reflect the updated 16-bit NWK address con-
5733 tained in the Device_ponce.

5734 The ZigBee End Device shall process the NLME-NWK-STATUS.indication sent from the NWK layer. If
5735 the error code equals to 0x09 (Parent Link Failure), the ZED will update its failure counter maintained in
5736 ZDO. If the value of the failure counter is smaller than the :Config_Parent_Link_Retry_Threshold attribute,
5737 the ZED may decide to issue further commands to attempt to communicate with the parent node, depending
5738 on the application of the ZED. If the value of the failure counter exceeds the :Config_Parent_Link_-
5739 Retry_Threshold attribute, the ZED shall then prepare to start the rejoin process. Note that implementers
5740 may optionally use a more accurate time-windowed scheme to identify a link failure.

5741 The rejoin process mirrors the MAC association process very closely, however, a device is permitted to re-
5742 join a parent that is not accepting new associations. The ZDO may use the
5743 NLME-NETWORK-DISCOVERY.

5744 request primitive to detect potential alternative parents, and in order to optimize recovery latency and reli-
5745 ability, shall select an appropriate new parent based on the following information from that device's beacon:

- 5746 • PAN ID
5747 • EPID (Extended PAN ID)
5748 • Channel
5749 • Signal strength
5750 • Whether the potential parent indicates that it is currently able to communicate with its Trust Center
5751 • Whether this device has recently failed to join this parent, or this network

5752 Once a potential parent has been selected, the ZDO shall issue an NLME-JOIN.request primitive with
5753 RejoinNetwork set to 0x02.

5754 The start time of the rejoin process is determined by the time the last NLME-JOIN.request primitive was
5755 sent and by the attribute :Config_Rejoin_Interval. Only if the interval between the current and the previous
5756 NLME-JOIN.request sent time is longer than the :Config_Rejoin_Interval shall a new NLME-JOIN.request
5757 primitive be sent. The application may want to gradually increase the :Config_Rejoin_Interval if a certain
5758 number of retries have been done (or a certain period of time has passed) but none of them were successful.
5759 The :Config_Rejoin_Interval should not exceed the :Config_Max_Rejoin_Interval. Every time an
5760 NLME-JOIN.confirm has been successfully received, the ZDO shall reset its failure counter to zero and the
5761 :Config_Rejoin_Interval attribute to its initial value. The choice of the default initial value and the algo-
5762 rithm of increasing the rejoin interval shall be determined by the application, and is out of the scope of this
5763 document.

5764 If the ZigBee End Device rejoins a new parent using the rejoin process, it shall complete the address con-
5765 flict process in section 3.6.1.9.

5766 **2.5.4.5.6 Support for Commissioning Applications**

5767 ZigBee devices in the field will need commissioning, and it will be up to developers to provide applications
5768 that perform such commissioning. There is a risk that applications from different vendors will work differ-
5769 ently, thereby diminishing the ability of ZigBee devices from different vendors to operate seamlessly on the
5770 same network. As a partial solution to this problem, this section lists a common set of configuration attrib-
5771 utes for ZigBee devices and outlines a common procedure for devices to use at start-up time. The other
5772 critical component of the solution is a common set of commissioning protocols and procedures, which are
5773 outside the scope of this document.

5774 **2.5.4.5.6.1 Configuration Attributes**

5775 The startup procedure outlined in section 2.5.4.5.6.2 is designed in such a way that, by using it consistently,
5776 devices can go through all the stages of commissioning up to being joined to the proper ZigBee network
5777 and able to send and receive application data traffic. Later-stage commissioning, including the commis-
5778 sioning of bindings and group membership is discussed briefly in section 2.5.4.5.6.3. The procedure makes
5779 use of the system attributes listed in Table 2.143.

5780

Table 2.143 Startup Attributes

Name	Reference	Comment
<i>nwkExtendedPANID</i>	Table 3.43	This is the extended PANID of the network to which the device is joined. If it has a value of 0x0000000000000000, then the device is not connected to a network.
<i>apsDesignatedCoordinator</i>	Table 2.24	This boolean flag indicates whether the device should assume on startup that it must become a ZigBee coordinator.
<i>apsChannelMask</i>	Table 2.24	This is the mask containing allowable channels on which the device may attempt to form or join a network at startup time.
<i>apsUseExtendedPANID</i>	Table 2.24	The 64-bit identifier of the network to join or form.
<i>apsUseInsecureJoin</i>	Table 2.24	A Boolean flag that indicates if it is OK to use insecure join on startup.

5781 **2.5.4.5.6.2 Startup Procedure**

5782 The startup procedure uses the attributes listed in section 2.5.4.5.6.1 to perform a controlled startup of the
5783 ZigBee networking facilities of a device. The procedure should be run whenever the device restarts, but
5784 may also be run under application control at the discretion of the developer.

5785 When a device starts up, it should check the value of *nwkExtendedPANID*. If *nwkExtendedPANID* has a
5786 non-zero value, then the device should assume it has all the network parameters required to operate on a
5787 network. Note that the device should assume the channel identifier present in its current network parame-
5788 ters but may need to scan over the ChannelMask if the *nwkExtendedPANID* is not found. In order for this to
5789 work effectively across power failures and processor resets, *nwkExtendedPANID* must be placed in
5790 non-volatile storage.

5791 If the device finds it is not connected to a network, then it should check the value of
5792 *apsDesignatedCoordinator*. If this attribute has a value of TRUE, then the device should follow the proce-
5793 dures for starting a network outlined in section 3.6.1.4.1 and should use the value of *apsChannelMask* for
5794 the ScanChannels parameter of the NLME-NETWORK-FORMATION.request primitive, and set
5795 nwkExtendedPANID to the value given in *apsUseExtendedPANID* if *apsUseExtendedPANID* has a
5796 non-zero value.

5797 If the device is not the designated coordinator and *apsUseExtendedPANID* has a non-zero value, the device
5798 should attempt to rejoin the network specified in *apsUseExtendedPANID*. To do this, it should use
5799 NLME-JOIN.request with the ExtendedPANID parameter equal to the value of *apsUseExtendedPANID*,
5800 the ScanChannels parameter of the primitive equal to the value of the *apsChannelMask* configuration at-
5801 tribute. The RejoinNetwork parameter of the NLME-JOIN.request primitive should have a value of 0x02
5802 indicating rejoin.

5803 If the network rejoin attempt fails, and the value of the *apsUseInsecureJoin* attribute of the AIB has a value
5804 of TRUE, then the device should follow the procedure outlined in section 3.6.1.4.1 for joining a network,
5805 using *apsChannelMask* any place that a ScanChannels mask is called for. If *apsUseExtendedPANID* has a
5806 non-zero value, then the device should join only the specified network and the procedure should fail if that
5807 network is found to be inaccessible. If *apsUseExtendedPANID* is equal to 0x0000000000000000, then the
5808 device should join the best available network.

5809 **2.5.4.5.6.3 Further Commissioning**

5810 Once a device is on a network and capable of communicating with other devices on the network in a secure
5811 manner, other commissioning becomes possible. Other items that should be subject to commissioning are
5812 shown in Table 2.144.

5813 **Table 2.144 Additional Commissioning Attributes**

Name	Reference	Comment
<i>apsBindingTable</i>	Table 2.24	The binding table for this device. Binding provides a separation of concerns in the sense that applications may operate without having to manage recipient address information for the frames they emit. This information can be input at commissioning time without the main application on the device even being aware of it.
<i>nwkGroupIDTable</i>	Table 3.43	Commissioning applications should be able to manage group membership of a device and its endpoints by accessing this table.
<i>nwkSecurityMaterialSet</i>	Table 4.2	This set contains the network keying material, which should be accessible to commissioning applications.
<i>apsDeviceKeyPairSet</i>	Table 4.38	This is the set of link key pairs for devices that it wants to communicate using application layer encryption.
<i>apsTrustCenterAddress</i>	Table 4.38	The IEEE address of the Trust Center.

Name	Reference	Comment
<i>nwkNetworkAddress</i>	Table 3.44	Commissioning applications may set the network short address of devices as long as address conflicts that may arise as a result are subject to address conflict resolution as described in section 3.6.1.9.

5814 **2.5.4.6 Device and Service Discovery**

5815 The Device and Service Discovery function supports:

- 5816 • Device Discovery
5817 • Service Discovery

5818 Device Management performs the above functions with the ZigBee Device Profile (see clause 2.4).

5819 **2.5.4.6.1 Optional and Mandatory Attributes Within Device and Service Dis-** 5820 **cove**

5821 All of the request attributes within the Device and Service Discovery Object are optional for all ZigBee
5822 logical device types. The responses listed in Table 2.145 as mandatory are mandatory for all ZigBee logical
5823 device types, and the responses listed as optional are optional for all ZigBee logical device types. See sec-
5824 tion The ZigBee Device Profile2.4 for a description of any of these attributes.

5825 **Table 2.145 Device and Service Discovery Attributes**

Attribute	M/O	Type
NWK_addr_req	O	Public
NWK_addr_rsp	M	Public
IEEE_addr_req	O	Public
IEEE_addr_rsp	M	Public
Node_Desc_req	O	Public
Node_Desc_rsp	M	Public
Power_Desc_req	O	Public
Power_Desc_rsp	M	Public
Simple_Desc_req	O	Public
Simple_Desc_rsp	M	Public
Active_EP_req	O	Public

Attribute	M/O	Type
Active_EP_rsp	M	Public
Match_Desc_req	O	Public
Match_Desc_rsp	M	Public
Complex_Desc_req	O	Public
Complex_Desc_rsp	O	Public
User_Desc_req	O	Public
User_Desc_rsp	O	Public
Device_annce	M	Public
Parent_annce	M	Public
Parent_annce_rsp	M	Public
User_Desc_set	O	Public
User_Desc_conf	O	Public
Sys- tem_Server_Discovery_req	O	Public
Sys- tem_Server_Discovery_rsp	O	Public
Discovery_Cache_req	O	Public
Discovery_Cache_rsp	O	Public
Discovery_store_req	O	Public
Discovery_store_rsp	O	Public
Node_Desc_store_req	O	Public
Node_Desc_store_rsp	O	Public
Power_Desc_store_req	O	Public
Power_Desc_store_rsp	O	Public

Attribute	M/O	Type
Active_EP_store_req	O	Public
Active_EP_store_rsp	O	Public
Simple_Desc_store_req	O	Public
Simple_Desc_store_rsp	O	Public
Remove_node_cache_req	O	Public
Remove_node_cache_rsp	O	Public
Find_node_cache_req	O	Public
Find_node_cache_rsp	O	Public

5826 2.5.4.7 Security Manager

5827 The security manager determines whether security is enabled or disabled and, if enabled, shall perform the
5828 following:

- 5829 • Establish Key
- 5830 • Transport Key
- 5831 • Authentication

5832 2.5.4.7.1 Optional and Mandatory Attributes Within Security Manager

5833 The Security Manager itself is an optional object for all ZigBee Device Types. If the Security Manager is
5834 present, all requests and responses are mandatory for all ZigBee device types. If the Security Manager is
5835 not present, none of the attributes in the Security Manager are present for any ZigBee logical device type.
5836 See section 2.4 for a description of any of the primitives listed in Table 2.146.

5837 **Table 2.146 Security Manager Attributes**

Attribute	M/O	Type
APSME-TRANSPORT-KEY.request	O	Public
APSME-TRANSPORT-KEY.indication	O	Public
APSME-UPDATE-DEVICE.request	O	Public
APSME-UPDATE-DEVICE.indication	O	Public
APSME-REMOVE-DEVICE.request	O	Public
APSME-REMOVE-DEVICE.indication	O	Public

Attribute	M/O	Type
APSME-REQUEST-KEY.request	O	Public
APSME-REQUEST-KEY.indication	O	Public
APSME-SWITCH-KEY.request	O	Public
APSME-SWITCH-KEY.indication	O	Public

5838 **2.5.4.8 Binding Manager**

5839 The Binding Management function supports:

- 5840 • End Device Binding
5841 • Bind and Unbind

5842 Binding Management performs the above functions with ZigBee Device Profile commands plus
5843 APSME-SAP primitives to commit/remove binding table entries once the indication arrives on the ZigBee
5844 coordinator, router, or end device supporting the binding table.

5845 **2.5.4.8.1 Optional and Mandatory Attributes Within Binding Manager**

5846 The Binding Manager is an optional object for all ZigBee Device Types.

5847 If the Binding Manager is present, all requests are optional for all ZigBee logical device types. Responses
5848 shall be supported on devices which implement a binding table cache, and on devices which correspond to
5849 the source address for the binding table entries held on those devices.

5850 If the Binding Manager is not present, all requests and all responses for all ZigBee logical device types
5851 shall not be supported. Table 2.147 summarizes Binding Manager attributes.

Table 2.147 Binding Manager Attributes

Attribute	M/O	Type
End_Device_Bind_req	O	Public
End_Device_Bind_rsp	O	Public
Bind_req	O	Public
Bind_rsp	O	Public
Unbind_req	O	Public
Unbind_rsp	O	Public
Bind_Register_req	O	Public
Bind_Register_rsp	O	Public
Replace_Device_req	O	Public
Replace_Device_rsp	O	Public
Store_Bkup_Bind_Entry_req	O	Public
Store_Bkup_Bind_Entry_rsp	O	Public
Remove_Bkup_Bind_req	O	Public
Remove_Bkup_Bind_rsp	O	Public
Backup_Bind_Table_req	O	Public
Backup_Bind_Table_rsp	O	Public
Recover_Bind_Table_req	O	Public
Recover_Bind_Table_rsp	O	Public
Backup_Source_Bind_req	O	Public
Backup_Source_Bind_rsp	O	Public
Recover_Source_Bind_req	O	Public
Recover_Source_Bind_rsp	O	Public

Attribute	M/O	Type
APSME-BIND.request	O	Private
APSME-BIND.confirm	O	Private
APSME-UNBIND.request	O	Private
APSME-UNBIND.confirm	O	Private

2.5.4.9 Network Manager

The Network Management function supports:

- Network Discovery
- Network Formation
- Permit/Disable Associations
- Association and Disassociation
- Route Discovery
- Network Reset
- Radio Receiver State Enable/Disable
- Get and Set of Network Management Information Block Data
- Detecting and reporting interference
- Receive network interference reports and change network channels if the particular node is identified as the network manager for the overall PAN

Network Management performs the above functions with NLME-SAP primitives (see Chapter 3).

2.5.4.9.1 Optional and Mandatory Attributes Within Network Manager

The Network Manager is a mandatory object for all ZigBee Device Types.

The Network Discovery, Get, and Set attributes (both requests and confirms) are mandatory for all ZigBee logical device types.

If the ZigBee logical device type is ZigBee Coordinator, the NWK Formation request and confirm, the NWK Leave request, NWK Leave indication, NWK Leave confirm, NWK Join indication, NWK Permit Joining request, NWK Permit Joining confirm, NWK Route Discovery request, and NWK Route Discovery confirm shall be supported. The NWK Direct Join request and NWK Direct Join confirm may be supported. The NWK Join request and the NWK Join confirm shall not be supported.

If the ZigBee logical device type is ZigBee Router, the NWK Formation request and confirm shall not be supported except if forming distributed networks. Additionally, the NWK Start Router request, NWK Start Router confirm, NWK Join request, NWK Join confirm, NWK Join indication, NWK Leave request, NWK Leave confirm, NWK Leave indication, NWK Permit Joining request, NWK Permit Joining confirm, NWK Route Discovery request, and NWK Route Discovery confirm shall be supported. The NWK Direct Join request and NWK Direct Join confirm may be supported.

If the ZigBee logical device type is ZigBee End Device, the NWK Formation request and confirm plus the NWK Start Router request and confirm shall not be supported. Additionally, the NWK Join indication and NWK Permit Joining request shall not be supported. The NWK Join request, NWK Join confirm, NWK Leave request, NWK Leave indication, NWK Leave confirm shall be supported.

5886 For all ZigBee logical devices types, the NWK Sync request, indication and confirm plus NWK reset re-
5887 quest and confirm plus NWK route discovery request and confirm shall be optional. Table 2.148 summarizes Network Manager Attributes. See Chapter 3 for a description of any of the primitives listed in Table
5888 2.148.
5889

5890 For all ZigBee logical device types, reception of the NWK Network Status indication shall be supported,
5891 but no action is required in this version of the specification.

5892 **Table 2.148 Network Manager Attributes**

Attribute	M/O	Type
NLME-GET.request	M	Private
NLME-GET.confirm	M	Private
NLME-SET.request	M	Private
NLME-SET.confirm	M	Private
NLME-NETWORK-DISCOVERY.request	M	Public
NLME-NETWORK-DISCOVERY.confirm	M	Public
NLME-NETWORK-FORMATION.request	O	Private
NLME-NETWORK-FORMATION.confirm	O	Private
NLME-START-ROUTER.request	O	Private
NLME-START-ROUTER.confirm	O	Private
NLME-JOIN.request	O	Private
NLME-JOIN.confirm	O	Private
NLME-JOIN.indication	O	Private
NLME-PERMIT-JOINING.request	O	Public
NLME-PERMIT-JOINING.confirm	O	Public
NLME-DIRECT-JOIN.request	O	Public
NLME-DIRECT-JOIN.confirm	O	Public
NLME_LEAVE.request	M	Public
NLME_LEAVE.confirm	M	Public

Attribute	M/O	Type
NLME_LEAVE.indication	M	Public
NLME-RESET.request	O	Private
NLME-RESET.confirm	O	Private
NLME-SYNC.request	O	Public
NLME-SYNC.indication	O	Public
NLME-SYNC.confirm	O	Public
NLME-NWK-STATUS.indication	M	Private
NLME-ROUTE-DISCOVERY.request	O	Public
NLME-ROUTE-DISCOVERY.confirm	O	Private
NLME-ED-SCAN.request	O	Private
NLME-ED-SCAN.confirm	O	Private
NLME-START-BACKOFF.request	O	Private

5893

5894 A single device in the network can become the Network Channel Manager. The operation of the network
5895 channel manager is described in Annex E. All other devices in the network are responsible for tracking
5896 message delivery failures and reporting interference in accordance with Annex E.

5897 **2.5.4.10 Node Manager**

5898 The Node Manager supports the ability to request and respond to management functions. These manage-
5899 ment functions only provide visibility to external devices regarding the operating state of the device re-
5900 ceiving the request.

5901 **2.5.4.11 Group Manager**

5902 The Group Manager supports the ability to include application objects within groups or to remove applica-
5903 tion objects from groups. The group management functions operate only on application objects within the
5904 local device. Mechanisms to manage groups on other devices are beyond the scope of this document.

5905 **2.5.5 Configuration Attributes**

5906 This attribute is used to represent the minimum mandatory and/or optional attributes used as configura-
5907 tion attributes for a device.

Table 2.149 Configuration Attributes

Attribute	M/O	Type
:Config_Node_Descriptor	M	Public
:Config_Power_Descriptor	M	Public
:Config_Simple_Descriptors	M	Public
:Config_NWK_Scan_Attempts	M	Private
:Config_NWK_Time_btwn_Scans	M	Private
:Config_Complex_Descriptor	O	Public
:Config_User_Descriptor	O	Public
:Config_Max_Bind	O	Private
:Config_EndDev_Bind_Timeout	O	Private
:Config_Permit_Join_Duration	O	Public
:Config_NWK_Security_Level	O	Private
:Config_NWK_Secure_All_Frames	O	Private
:Config_NWK_Leave_removeChildren	O	Private
:Config_NWK_BroadcastDeliveryTime	O	Private
:Config_NWK_TransactionPersistenceTime	O	Private
:Config_NWK_indirectPollRate	O	Private
:Config_Max_Assoc	O	Private
:Config_NWK_Join_Direct_Addrs	O	Public
:Config_Parent_Link_Retry_Threshold	O	Public
:Config_Rejoin_Interval	O	Public
:Config_Max_Rejoin_Interval	O	Public

5909 **2.5.5.1 Configuration Attribute Definitions**

5910 **Table 2.150 Configuration Attribute Definitions**

Attribute	Description	When Updated
:Config_Node_Descriptor	Contents of the Node Descriptor for this device (see section 2.3.2.3).	The :Config_Node_Descriptor is either created when the application is first loaded or initialized with a commissioning tool prior to when the device begins operations in the network. It is used for service discovery to describe node features to external inquiring devices.
:Config_Power_Descriptor	Contents of the Power Descriptor for this device (see section 2.3.2.4).	The :Config_Power_Descriptor is either created when the application is first loaded or initialized with a commissioning tool prior to when the device begins operations in the network. It is used for service discovery to describe node power features to external inquiring devices.
:Config_Simple_Descriptors	Contents of the Simple Descriptor(s) for each active endpoint for this device (see section 2.3.2.5).	The :Config_Simple_Descriptors are created when the application is first loaded and are treated as “read-only.” The Simple Descriptor are used for service discovery to describe interfacing features to external inquiring devices.
:Config_NWK_Scan_Attempts	Integer value representing the number of scan attempts to make before the NWK layer decides which ZigBee coordinator or router to associate with (see section 2.5.4.5). This attribute has default value of 5 and valid values between 1 and 255.	The :Config_NWK_Scan_Attempts is employed within ZDO to call the NLME-NETWORK-DISCOVERY.req primitive the indicated number of times (for routers and end devices).

Attribute	Description	When Updated
:Config_NWK_Time_btwn_Scans	<p>Integer value representing the time duration (in OctetDurations) between each NWK discovery attempt described by :Config_NWK_Scan_Attempts (see section 2.5.4.5).</p> <p>This attribute has a default value of 0xc35 OctetDurations (100 milliseconds on 2.4GHz) and valid values between 1 and 0x1f3fe1 OctetDurations (65535 milliseconds on 2.4GHz).</p>	<p>The :Config_NWK_Time_btwn_Scans is employed within ZDO to provide a time duration between the NLME-NETWORK-DISCOVERY.request attempts.</p>
:Config_Complex_Descriptor	<p>Contents of the (optional) Complex Descriptor for this device (see section 2.3.2.6).</p>	<p>The :Config_Complex_Descriptor is either created when the application is first loaded or initialized with a commissioning tool prior to when the device begins operations in the network. It is used for service discovery to describe extended device features for external inquiring devices.</p>
:Config_User_Descriptor	<p>Contents of the (optional) User Descriptor for this device (see section 2.3.2.7).</p>	<p>The :Config_User_Descriptor is either created when the application is first loaded or initialized with a commissioning tool prior to when the device begins operations in the network. It is used for service discovery to provide a descriptive character string for this device to external inquiring devices.</p>
:Config_Max_Bind	<p>A constant which describes the maximum number of binding entries permitted.</p>	<p>The :Config_Max_Bind is a maximum number of supported Binding Table entries for this device.</p>
:Config_EndDev_Bind_Timeout	<p>Timeout value in seconds employed in End Device Binding (see section 2.4.3.2).</p>	<p>The :Config_EndDev_Bind_Timeout is employed only on ZigBee Coordinators and used to determine whether end device bind requests have been received within the timeout window.</p>

Attribute	Description	When Updated
:Config_Permit_Join_Duration	Permit Join Duration value set by the NLME-PERMIT-JOINING.req primitive (see Chapter 3).	The default value for :Config_Permit_Join_Duration is 0x00, however, this value can be established differently according to the needs of the profile.
:Config_NWK_Security_Level	Security level of the network (see Chapter 3).	This attribute is used only on the Trust Center and is used to set the level of security on the network.
:Config_NWK_Secure_All_Frames	If all network frames should be secured (see Chapter 3).	This attribute is used only on the Trust Center and is used to determine if network layer security shall be applied to all frames in the network.
:Config_NWK_Leave_removeChildren	Sets the policy as to whether child devices are to be removed if the device is asked to leave the network via NLME-LEAVE (see Chapter 3).	The policy for setting this attribute is found in the Stack Profile employed.
:Config_NWK_BroadcastDeliveryTime	See Chapter 3, Table 3-57.	The value for this configuration attribute is established in the Stack Profile.
:Config_NWK_TransactionPersistenceTime	See Table 3-44. This attribute is mandatory for the ZigBee coordinator and ZigBee routers and not used for ZigBee End Devices.	The value for this configuration attribute is established in the Stack Profile.
:Config_NWK_Alt_protocol_version	Sets the list of protocol version numbers, other than the current protocol version number, that the device may choose to employ in a PAN that it joins. This attribute is applicable only to ZigBee routers or end devices. The protocol version numbers in the list must refer to older versions of the ZigBee Specification.	:Config_NWK_Alt_protocol_version permits ZigBee routers and ZigBee end devices to join networks discovered that employ an earlier version of the ZigBee Specification; Since this attribute is optional, devices may also be created omitting this attribute which require only the current version of the ZigBee Specification; This attribute would be omitted in cases where certain features are required that are contained only in the current specification or where code size is limited in the device.

Attribute	Description	When Updated
:Config_NWK_indirectPollRate	Sets the poll rate, in milliseconds, for the device to request indirect transmission messages from the parent.	The value for this configuration attribute is established by the application profile deployed on the device.
:Config_Max_Assoc	Sets the maximum allowed associations, either of routers, end devices, or both, to a parent router or coordinator.	The value for this configuration attribute is established by the stack profile in use on the device. Note that for some stack profiles, the maximum associations may have a dimension which provides for separate maximums for router associations and end device associations.
:Config_NWK_Join_Direct_Addrs	<p>Consists of the following fields:</p> <ul style="list-style-type: none"> DeviceAddress - 64-bit IEEE address for the device to be direct joined CapabilityInformation - Operating capabilities of the device to be direct joined Link Key- If security is enabled, link key for use in the key-pair descriptor for this new device (see Table 4.39) See section 3.2.2.14 for details. 	:Config_NWK_Join_Direct_Addrs permits the ZigBee Coordinator or Router to be pre-configured with a list of addresses to be direct joined.
:Config_Parent_Link_Retry_Threshold	Contents of the link retry threshold for parent link (see section 2.5.4.5.5.2)	The :Config_Parent_Link_Retry_Threshold is either created when the application is first loaded or initialized with a commissioning tool. It is used for the ZED to decide how many times it should retry to connect to the parent router before initiating the rejoin process.
:Config_Rejoin_Interval	Contents of the rejoin interval (see section 2.5.4.5.5.2).	The :Config_Rejoin_Interval is either created when the application is first loaded or initialized with a commissioning tool. It is used by the ZED to decide how often it should initiate the rejoin process.

Attribute	Description	When Updated
:Config_MAX_Rejoin_Interval	Contents of the maximal rejoin interval (see section 2.5.4.5.5.2).	The :Config_MAX_Rejoin_Interval is either created when the application is first loaded or initialized with a commissioning tool. It is used by the ZED to set the maximum value permitted for :Config_Rejoin_Interval during the rejoin procedure.

5911

5912

5913
5914
5915
5916
5917
5918
5919
5920
5921
5922
5923
5924
5925
5926
5927
5928
5929
5930
5931
5932
5933
5934

This page intentionally left blank.

5935

5936

CHAPTER 3 NETWORK SPECIFICATION

5937

3.1 General Description

5938

3.1.1 Network (NWK) Layer Overview

5939

5940

5941

5942

5943

5944

5945

5946

The network layer is required to provide functionality to ensure correct operation of the IEEE 802.15.4 MAC sub-layer and to provide a suitable service interface to the application layer. To interface with the application layer, the network layer conceptually includes two service entities that provide the necessary functionality. These service entities are the data service and the management service. The NWK layer data entity (NLDE) provides the data transmission service via its associated SAP, the NLDE-SAP, and the NWK layer management entity (NLME) provides the management service via its associated SAP, the NLME-SAP. The NLME utilizes the NLDE to achieve some of its management tasks and it also maintains a database of managed objects known as the network information base (NIB).

5947

3.1.2 Network Layer Data Entity (NLDE)

5948

5949

The NLDE shall provide a data service to allow an application to transport application protocol data units (APDU) between two or more devices. The devices themselves must be located on the same network.

5950

The NLDE will provide the following services:

5951

5952

- **Generation of the Network level PDU (NPDU):** The NLDE shall be capable of generating an NPDU from an application support sub-layer PDU through the addition of an appropriate protocol header.
- **Topology-specific routing:** The NLDE shall be able to transmit an NPDU to an appropriate device that is either the final destination of the communication or the next step toward the final destination in the communication chain.
- **Security:** The ability to ensure both the authenticity and confidentiality of a transmission.

5957

3.1.2.1 Network Layer Management Entity (NLME)

5958

The NLME shall provide a management service to allow an application to interact with the stack.

5959

The NLME shall provide the following services:

5960

5961

5962

- **Configuring a new device:** this is the ability to sufficiently configure the stack for operation as required. Configuration options include beginning an operation as a ZigBee coordinator or joining an existing network.
- **Starting a network:** this is the ability to establish a new network.
- **Joining, rejoining and leaving a network:** this is the ability to join, rejoin or leave a network as well as the ability of a ZigBee coordinator or ZigBee router to request that a device leave the network.
- **Addressing:** this is the ability of ZigBee coordinators and routers to assign addresses to devices joining the network.

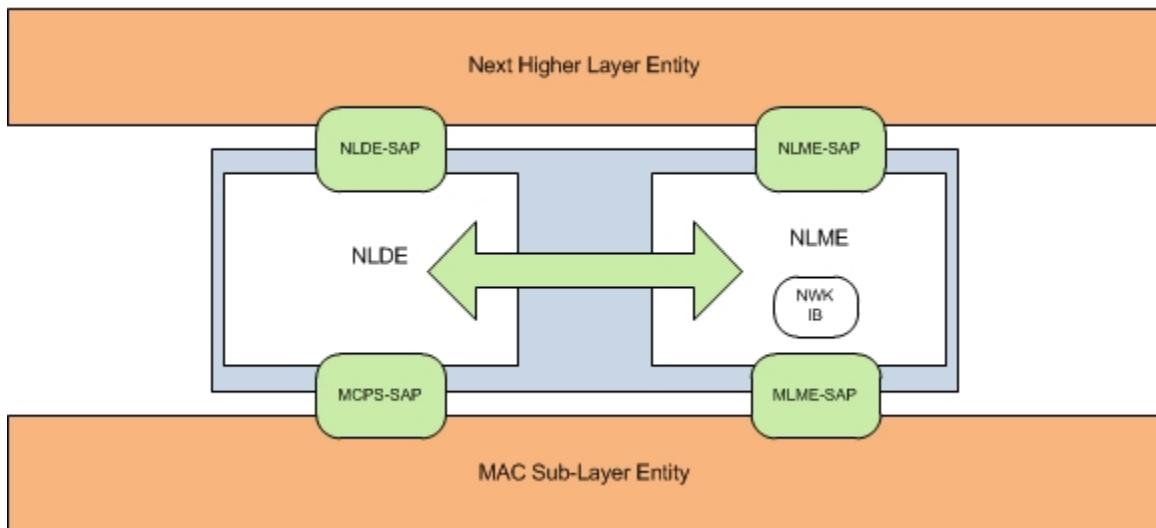
- 5968
- **Neighbor discovery:** this is the ability to discover, record, and report information pertaining to the one-hop neighbors of a device.
 - **Route discovery:** this is the ability to discover and record paths through the network, whereby messages may be efficiently routed.
 - **Reception control:** this is the ability for a device to control when the receiver is activated and for how long, enabling MAC sub-layer synchronization or direct reception.
 - **Routing:** this is the ability to use different routing mechanisms such as unicast, broadcast, multicast or many to one to efficiently exchange data in the network.
- 5970

5976 3.2 Service Specification

5977 Figure 3.1 depicts the components and interfaces of the NWK layer.

5978 The NWK layer provides two services, accessed through two service access points (SAPs). These are the
5979 NWK data service, accessed through the NWK layer data entity SAP (NLDE-SAP), and the NWK management
5980 service, accessed through the NWK layer management entity SAP (NLME-SAP). These two services provide the interface
5981 between the application and the MAC sub-layer, via the MCPS-SAP and
5982 MLME-SAP interfaces (See [B1]). In addition to these external interfaces, there is also an implicit interface
5983 between the NLME and the NLDE that allows the NLME to use the NWK data service.

5984 **Figure 3.1 The NWK Layer Reference Model**



5986 3.2.1 NWK Data Service

5987 The NWK layer data entity SAP (NLDE-SAP) supports the transport of application protocol data units
5988 (APDUs) between peer application entities. Table 3.1 lists the primitives supported by the NLDE-SAP and
5989 the sections in which these primitives are discussed.

5990 **Table 3.1 NLDE-SAP Primitives**

NLDE-SAP Primitive	Request	Confirm	Indication
NLDE-DATA	3.2.1.1	3.2.1.2	3.2.1.3

5991 **3.2.1.1 NLDE-DATA.request**

5992 This primitive requests the transfer of a data PDU (NSDU) from the local APS sub-layer entity to a single
5993 or multiple peer APS sub-layer entities.

5994 **3.2.1.1.1 Semantics of the Service Primitive**

5995 The semantics of this primitive are as follows:

5996 NLDE-DATA.request {
5997 DstAddrMode,
5998 DstAddr,
5999 NsduLength,
6000 Nsdu,
6001 NsduHandle,
6002 UseAlias,
6003 AliasSrcAddr,
6004 AliasSeqNumber,
6005 Radius,
6006 NonmemberRadius,
6007 DiscoverRoute,
6008 SecurityEnable
6009 }

6010
6011 Table 3.2 specifies the parameters for the NLDE-DATA.request primitive. Support of the additional pa-
6012 rameters UseAlias, AliasSrcAddr, AliasSeqNumb in the NLDE-DATA.request primitive is required if GP
6013 feature is to be supported by the implementation.

6014 **Table 3.2 NLDE-DATA.request Parameters**

Name	Type	Valid Range	Description
DstAddrMode	Integer	0x01 or 0x02	The type of destination address supplied by the DstAddr parameter. This may have one of the following two values: 0x01=16-bit multicast group address 0x02=16-bit network address of a device or a 16-bit broadcast address
DstAddr	16-bit Address	0x0000-0xffff	Destination address.

Name	Type	Valid Range	Description
NsduLength	Integer	0 to $aMaxPHYPacketSize - (nwkcMACFrameOverhead + nwkcMinHeaderOverhead)$	The number of octets comprising the NSDU to be transferred.
Nsdu	Set of Octets	-	The set of octets comprising the NSDU to be transferred.
NsduHandle	Integer	0x00 – 0xff	The handle associated with the NSDU to be transmitted by the NWK layer entity.
UseAlias	Boolean	TRUE or FALSE	<p>The next higher layer MAY use the <i>UseAlias</i> parameter to request alias usage by NWK layer for the current frame. If the <i>UseAlias</i> parameter has a value of FALSE, meaning no alias usage,</p> <p>then the parameters <i>AliasSrcAddr</i> and <i>AliasSeqNumb</i> will be ignored.</p> <p>Otherwise, a value of TRUE denotes that the values supplied in <i>AliasSrcAddr</i> and <i>AliasSeqNumb</i> are to be used.</p>
AliasSrcAddr	16-bit address	Any valid device address except a broadcast address	The source address to be used for this NSDU. If the <i>UseAlias</i> parameter has a value of FALSE, the <i>AliasSrcAddr</i> parameter is ignored.
AliasSeqNumb	integer	0x00-0xff	The sequence number to be used for this NSDU. If the <i>UseAlias</i> parameter has a value of FALSE, the <i>AliasSeqNumb</i> parameter is ignored.
Radius	Unsigned Integer	0x00 – 0xff	The distance, in hops, that a frame will be allowed to travel through the network.
NonmemberRadius	Integer	0x00 – 0x07	The distance, in hops, that a multicast frame will be relayed by nodes not a member of the group. A value of 0x07 is treated as infinity.

Name	Type	Valid Range	Description
DiscoverRoute	Integer	0x00 – 0x01	The DiscoverRoute parameter may be used to control route discovery operations for the transit of this frame (see section 3.6.3.5): 0x00 = suppress route discovery 0x01 = enable route discovery
SecurityEnable	Boolean	TRUE or FALSE	The SecurityEnable parameter may be used to enable NWK layer security processing for the current frame. If the <i>nwkSecurityLevel</i> attribute of the NIB has a value of 0, meaning no security, then this parameter will be ignored. Otherwise, a value of TRUE denotes that the security processing specified by the security level will be applied, and a value of FALSE denotes that no security processing will be applied.

3.2.1.1.2 When Generated

This primitive is generated by a local APS sub-layer entity whenever a data PDU (NSDU) is to be transferred to a peer APS sub-layer entity.

3.2.1.1.3 Effect on Receipt

If this primitive is received on a device that is not currently associated, the NWK layer will issue an NLDE-DATA.confirm primitive with a status of INVALID_REQUEST.

On receipt of this primitive, the NLDE first constructs an NPDU in order to transmit the supplied NSDU. If, during processing, the NLDE issues the NLDE-DATA.confirm primitive prior to transmission of the NSDU, all further processing is aborted. In constructing the new NPDU, the destination address field of the NWK header will be set to the value provided in the DstAddr parameter. If the UseAlias parameter has a value of TRUE, the source address field of the NWK header of the frame will be set to the value provided in the AliasSrcAddr parameter. If the UseAlias parameter has a value of FALSE, then the source address field will have the value of the *macShortAddress* attribute in the MAC PIB. The discover route sub-field of the frame control field of the NWK header will be set to the value provided in the DiscoverRoute parameter. If the supplied Radius parameter does not have a value of zero, then the radius field of the NWK header will be set to the value of the Radius parameter. If the Radius parameter has a value of zero, then the radius field of the NWK header will be set to twice the value of the *nwkMaxDepth* attribute of the NIB. If the UseAlias parameter has a value of TRUE, the sequence number field of the NWK header of the frame will be set to the value provided in the AliasSeqNumb parameter. If the UseAlias parameter has a value of FALSE, then the NWK layer will generate a sequence number for the frame as described in section 3.6.2.1 and the sequence number field of the NWK header of the frame will be set to this sequence number value. The multicast flag field of the NWK header will be set according to the value of the DstAddrMode parameter. If the DstAddrMode parameter has a value of 0x01, the NWK header will contain a multicast control field whose fields will be set as follows:

- The multicast mode field will be set to 0x01 if this node is a member of the group specified in the DstAddr parameter.
- Otherwise, the multicast mode field will be set to 0x00.
- The non-member radius and the max non-member radius fields will be set to the value of the Non-memberRadius parameter.

Once the NPDU is constructed, the NSDU is routed using the procedure described in section Upon Receipt of a Unicast Frame if it is a unicast, section 3.6.5 if it is a broadcast, or section 3.6.6.2 if it is a multicast. When the routing procedure specifies that the NSDU is to be transmitted, this is accomplished by issuing the MCPS-DATA.request primitive with both the SrcAddrMode and DstAddrMode parameters set to 0x02, indicating the use of 16-bit network addresses. The SrcPANId and DstPANId parameters should be set to the current value of *macPANId* from the MAC PIB. The SrcAddr parameter will be set to the value of *macShortAddr* from the MAC PIB. The value of the DstAddr parameter is the next hop address determined by the routing procedure. If the message is a unicast, bit b0 of the TxOptions parameter should be set to 1 denoting that an acknowledgement is required. On receipt of the MCPS-DATA.confirm primitive on a unicast, the NLDE issues the NLDE-DATA.confirm primitive with a status equal to that received from the MAC sub-layer. Upon transmission of a MCPS-DATA.confirm primitive, in the case of a broadcast or multicast, the NLDE immediately issues the NLDE-DATA.confirm primitive with a status of success.

If the *nwkSecurityLevel* NIB attribute has a non-zero value and the SecurityEnable parameter has a value of TRUE, then NWK layer security processing will be applied to the frame before transmission as described in clause 4.3. Otherwise, no security processing will be performed at the NWK layer for this frame. The security processing SHALL always be performed using device's own extended 64-bit IEEE address and Outgoing Frame Counter attribute of the NIB, and those values SHALL be put into the auxiliary NWK header of the frame, even if UseAlias parameter has a value of TRUE. If security processing is performed and it fails for any reason, then the frame is discarded and the NLDE issues the NLDE-DATA.confirm primitive with a Status parameter value equal to that returned by the security suite.

3.2.1.2 NLDE-DATA.confirm

This primitive reports the results of a request to transfer a data PDU (NSDU) from a local APS sub-layer entity to a single peer APS sub-layer entity.

3.2.1.2.1 Semantics of the Service Primitive

The semantics of this primitive are as follows:

NLDE-DATA.confirm	{
	Status
	NsduHandle,
	TxTime
	}

Table 3.3 specifies the parameters for the NLDE-DATA.confirm primitive.

6076

Table 3.3 NLDE-DATA.confirm Parameters

Name	Type	Valid Range	Description
Status	Status	INVALID_REQUEST, MAX_FRM_COUNTER, NO_KEY, BAD_CCM_OUTPUT, ROUTE_ERROR, BT_TABLE_FULL, FRAME_NOT_BUFFERED or any status values returned from security suite or the MCPS-DATA.confirm primitive (see [B1])	The status of the corresponding request.
NsduHandle	Integer	0x00 – 0xff	The handle associated with the NSDU being confirmed.
TxTime	Integer	Implementation specific	A time indication for the transmitted packet based on the local clock. The time should be based on the same point for each transmitted packet in a given im- plementation. This value is only provided if <i>nwkTimeStamp</i> is set to TRUE.

6077

3.2.1.2.2 When Generated

6078
6079

This primitive is generated by the local NLDE in response to the reception of an NLDE-DATA.request primitive.

6080

The Status field will reflect the status of the corresponding request, as described in section 3.2.1.1.3.

6081

3.2.1.2.3 Effect on Receipt

6082
6083
6084

On receipt of this primitive, the APS sub-layer of the initiating device is notified of the result of its request to transmit. If the transmission attempt was successful, the Status parameter will be set to SUCCESS. Otherwise, the Status parameter will indicate the error.

6085

3.2.1.3 NLDE-DATA.indication

6086
6087

This primitive indicates the transfer of a data PDU (NSDU) from the NWK layer to the local APS sub-layer entity.

6088 **3.2.1.3.1 Semantics of the Service Primitive**

6089 The semantics of this primitive are as follows:

6090 NLDE-DATA.indication {
6091 DstAddrMode,
6092 DstAddr,
6093 SrcAddr,
6094 NsduLength,
6095 Nsdu,
6096 LinkQuality
6097 RxTime
6098 SecurityUse
6099 }

6100

6101 Table 3.4 specifies the parameters for the NLDE-DATA.indication primitive.

6102 **Table 3.4 NLDE-DATA.indication Parameters**

Name	Type	Valid Range	Description
DstAddrMode	Integer	0x01 or 0x02	The type of destination address supplied by the DstAddr parameter. This may have one of the following two values: 0x01=16-bit multicast group address 0x02=16-bit network address of a device or a 16-bit broadcast address
DstAddr	16-bit Address	0x0000-0xffff	The destination address to which the NSDU was sent.
SrcAddr	16-bit Device address	Any valid device address except a broadcast address	The individual device address from which the NSDU originated.
NsduLength	Integer	0 to $aMaxPHYPacketSize - (nwkcMACFrameOverhead + nwkcMinHeaderOverhead)$	The number of octets comprising the NSDU being indicated.
Nsdu	Set of octets	—	The set of octets comprising the NSDU being indicated.

Name	Type	Valid Range	Description
LinkQuality	Integer	0x00 – 0xff	The link quality indication delivered by the MAC on receipt of this frame as a parameter of the MCPS-DATA.indication primitive (see [B1]).
RxTime	Integer	Implementation specific	A time indication for the received packet based on the local clock. The time should be based on the same point for each received packet on a given implementation. This value is only provided if <i>nwk-TimeStamp</i> is set to TRUE.
SecurityUse	Boolean	TRUE or FALSE	An indication of whether the received data frame is using security. This value is set to TRUE if security was applied to the received frame or FALSE if the received frame was unsecured.

6103

6104 3.2.1.3.2 When Generated

6105 This primitive is generated by the NLDE and issued to the APS sub-layer on receipt of an appropriately
6106 addressed data frame from the local MAC sub-layer entity.

6107 3.2.1.3.3 Effect on Receipt

6108 On receipt of this primitive, the APS sub-layer is notified of the arrival of data at the device.

6109 3.2.2 NWK Management Service

6110 The NWK layer management entity SAP (NLME-SAP) allows the transport of management commands
6111 between the next higher layer and the NLME. Table 3.5 lists the primitives supported by the NLME
6112 through the NLME-SAP interface and the sections containing details on each of these primitives.

6113 Table 3.5 Summary of the Primitives Accessed Through the NLME-SAP

Name	Section Number in this Specification			
	Request	Indication	Response	Confirm
NLME-NETWORK-DISCOVERY	3.2.1.1			3.2.2.2
NLME-NETWORK-FORMATION	3.2.2.3			3.2.2.4
NLME-PERMIT-JOINING	3.2.2.5			3.2.2.6
NLME-START-ROUTER	3.2.2.7			3.2.2.8

Name	Section Number in this Specification			
	Request	Indication	Response	Confirm
NLME-ED-SCAN	3.2.2.9			3.2.2.10
NLME-JOIN	3.2.2.11	3.2.2.12		3.2.2.13
NLME-DIRECT-JOIN	3.2.2.14			3.2.2.15
NLME-LEAVE	3.2.2.16	3.2.2.17		3.2.2.18
NLME-RESET	3.2.2.19			3.2.2.20
NLME-SYNC	3.2.2.22			3.2.2.24
NLME-SYNC-LOSS		3.2.2.23		
NLME-GET	3.2.2.26			3.2.2.27
NLME-SET	3.2.2.28			3.2.2.29
NLME-NWK-STATUS		3.2.2.30		
NLME-ROUTE-DISCOVERY	3.2.2.32			3.2.2.32

3.2.2.1 NLME-NETWORK-DISCOVERY.request

This primitive allows the next higher layer to request that the NWK layer discover networks currently operating within the POS.

3.2.2.1.1 Semantics of the Service Primitive

The semantics of this primitive are as follows:

```
NLME-NETWORK-DISCOVERY.request {
    ScanChannels,
    ScanDuration
}
```

Table 3.6 specifies the parameters for the NLME-NETWORK-DISCOVERY.request primitive.

6125

Table 3.6 NLME-NETWORK-DISCOVERY.request Parameters

Name	Type	Valid Range	Description
ScanChannels	Bitmap	32-bit field	The five most significant bits (b27,..., b31) are reserved. The 27 least significant bits (b0, b1,..., b26) indicate which channels are to be scanned (1 = scan, 0 = do not scan) for each of the 27 valid channels (see [B1]).
ScanDuration	Integer	0x00 – 0x0e	A value used to calculate the length of time to spend scanning each channel: The time spent scanning each channel is ($aBaseSuperframeDuration * (2^n + 1)$) symbols, where n is the value of the ScanDuration parameter. For more information on MAC sub-layer scanning (see [B1]).

6126

6127 **3.2.2.1.2 When Generated**

6128 This primitive is generated by the next higher layer of a ZigBee device and issued to its NLME to request
6129 the discovery of networks operating within the device's personal operating space (POS).

6130 **3.2.2.1.3 Effect on Receipt**

6131 On receipt of this primitive, the NWK layer will attempt to discover networks operating within the device's
6132 POS by performing an active scan over the channels specified in the ScanChannels argument and using
6133 channel page zero, for the period specified in the ScanDuration parameter. The scan is performed by means
6134 of the MLME-SCAN.request primitive.

6135 On receipt of the MLME-SCAN.confirm primitive, the NLME issues the
6136 NLME-NETWORK-DISCOVERY.confirm primitive containing the information about the discovered
6137 networks with a Status parameter value equal to that returned with the MLME-SCAN.confirm.

6138 **3.2.2.2 NLME-NETWORK-DISCOVERY.confirm**

6139 This primitive reports the results of a network discovery operation.

6140 **3.2.2.2.1 Semantics of the Service Primitive**

6141 The semantics of this primitive are as follows:

NLME-NETWORK-DISCOVERY.confirm	{
	Status
	NetworkCount,
	NetworkDescriptor,
	}

6147

6148 Table 3.7 describes the arguments of the NLME-NETWORK-DISCOVERY.confirm primitive.

Table 3.7 NLME-NETWORK-DISCOVERY.confirm Parameters

Name	Type	Valid Range	Description
Status	Status	Any status value returned with the MLME-SCAN.confirm primitive.	See [B1].
NetworkCount	Integer	0x00 – 0xff	Gives the number of networks discovered by the search.
NetworkDescriptor	List of network descriptors	The list contains the number of elements given by the NetworkCount parameter	A list of descriptors, one for each of the networks discovered. Table 3.8 gives a detailed account of the contents of each item.

6151 Table 3.8 gives a detailed account of the contents of a network descriptor from the NetworkDescriptor parameter.
6152

Table 3.8 Network Descriptor Information Fields

Name	Type	Valid Range	Description
ExtendedPANId	Integer	0x0000000000000001 - 0xfffffffffffffe	The 64-bit PAN identifier of the network.
LogicalChannel	Integer	Selected from the available logical channels supported by the PHY (see [B1]).	The current logical channel occupied by the network.
StackProfile	Integer	0x00 – 0x0f	A ZigBee stack profile identifier indicating the stack profile in use in the discovered network.
ZigBeeVersion	Integer	0x00 – 0x0f	The version of the ZigBee protocol in use in the discovered network.
BeaconOrder	Integer	0x00 – 0x0f	This specifies how often the MAC sub-layer beacon is to be transmitted by a given device on the network. For a discussion of MAC sub-layer beacon order see [B1].
SuperframeOrder	Integer	0x00 – 0x0f	For beacon-oriented networks, that is, beacon order < 15, this specifies the length of the active period of the superframe. For a discussion of MAC sub-layer superframe order

Name	Type	Valid Range	Description	
		see [B1].		
PermitJoining	Boolean	TRUE or FALSE	A value of TRUE indicates that at least one ZigBee router on the network currently permits joining, <i>i.e.</i> its NWK has been issued an NLME-PERMIT-JOINING primitive and, the time limit if given, has not yet expired.	
RouterCapacity	Boolean	TRUE or FALSE	This value is set to true if the device is capable of accepting join requests from router-capable devices and set to FALSE otherwise.	
EndDeviceCapacity	Boolean	TRUE or FALSE	This value is set to true if the device is capable of accepting join requests from end devices and set to FALSE otherwise.	

6154

3.2.2.2.2 When Generated

6155
6156

This primitive is generated by the NLME and issued to its next higher layer on completion of the discovery task initiated by an NLME-NETWORK-DISCOVERY.request primitive.

6157

3.2.2.2.3 Effect on Receipt

6158

On receipt of this primitive, the next higher layer is notified of the results of a network search.

6159

3.2.2.3 NLME-NETWORK-FORMATION.request

6160
6161

This primitive allows the next higher layer to request that the device start a new ZigBee network with itself as the coordinator and subsequently make changes to its superframe configuration.

6162 **3.2.2.3.1 Semantics of the Service Primitive**

6163 The semantics of this primitive are as follows:

6164 NLME-NETWORK-FORMATION.request {
6165 ScanChannels,
6166 ScanDuration,
6167 BeaconOrder,
6168 SuperframeOrder,
6169 BatteryLifeExtension
6170 DistributedNetwork
6171 DistributedNetworkAddress
6172 }

6173
6174 Table 3.9 specifies the parameters for the NLME-NETWORK-FORMATION.request primitive.

6175 **Table 3.9 NLME-NETWORK-FORMATION.request Parameters**

Name	Type	Valid Range	Description
ScanChannels	Bitmap	32-bit field	The five most significant bits (b27,..., b31) are reserved. The 27 least significant bits (b0, b1,... b26) indicate which channels are to be scanned in preparation for starting a network (1=scan, 0=do not scan) for each of the 27 valid channels (see [B1]).
ScanDuration	Integer	0x00 – 0x0e	A value used to calculate the length of time to spend scanning each channel. The time spent scanning each channel is $(aBaseSuperframeDuration * (2n + 1))$ symbols, where n is the value of the ScanDuration parameter (see [B1]).
BeaconOrder	Integer	0x00 – 0x0f	The beacon order of the network that the higher layers wish to form.
SuperframeOrder	Integer	0x00 – 0x0f	The superframe order of the network that the higher layers wish to form.
BatteryLifeExtension	Boolean	TRUE or FALSE	If this value is TRUE, the NLME will request that the ZigBee coordinator is started supporting battery life extension mode; If this value is FALSE, the NLME will request that the ZigBee coordinator is started without supporting battery life extension mode.

DistributedNetwork	Boolean	TRUE or FALSE	If this value is TRUE then it indicates that distributed network security will be used and therefore it is permissible for a ZigBee router to form the network. If FALSE, then this primitive may only be called by the ZigBee Coordinator.
DistributedNetworkAddress	Integer	0x0001 – 0xFFFF	The address the device will use when forming a distributed network.

6176 **3.2.2.3.2 When Generated**

6177 This primitive is generated by the next higher layer of a ZigBee coordinator-capable device and issued to
6178 its NLME to request the initialization of itself as the ZigBee coordinator of a new network.

6179 **3.2.2.3.3 Effect on Receipt**

6180 If DistributedNetwork is set to FALSE and the device is not capable of being a ZigBee coordinator, the
6181 NLME issues the NLME-NETWORK-FORMATION.confirm primitive with the Status parameter set to
6182 INVALID_REQUEST. If DistributedNetwork is set to TRUE and the device is not capable of being a
6183 ZigBee router then NLME issues the NLME-NETWORK-FORMATION.confirm primitive with the Status
6184 parameter set to INVALID_REQUEST. If DistributedNetwork is set to TRUE and the DistributedNet-
6185 workAddress is outside the valid range then processing shall fail with the Status parameter set to INVA-
6186 LID_REQUEST.

6187 The NLME requests that the MAC sub-layer first perform an energy detection scan and then an active scan
6188 on the specified set of channels and using channel page zero. To do this, the NLME issues the
6189 MLME-SCAN.request primitive to the MAC sub-layer with the ScanType parameter set to indicate an en-
6190 ergy detection scan and then issues the primitive again with the ScanType parameter set to indicate an ac-
6191 tive scan. After the completion of the active scan, on receipt of the MLME-SCAN.confirm primitive from
6192 the MAC sub-layer, the NLME selects a suitable channel. The NWK layer will pick a PAN identifier that
6193 does not conflict with that of any network known to be operating on the chosen channel. Once a suitable
6194 channel and PAN identifier are found, the NLME will choose an address as follows. If DistributedNet-
6195 work is set to FALSE, it shall use 0x0000 as the 16-bit short MAC address. If DistributedNetwork is set
6196 to TRUE then it shall use DistributedNetworkAddress as the 16-bit short MAC address. It shall inform
6197 the MAC sub-layer of the newly chosen address. To do this, the NLME issues the MLME-SET.request
6198 primitive to the MAC sub-layer to set the MAC PIB attribute *macShortAddress*. If the NIB attribute
6199 *nwkExtendedPANId* is equal to 0x0000000000000000, this attribute will be initialized with the value of the
6200 MAC constant *macExtendedAddress*. If no suitable channel or PAN identifier can be found, the NLME is-
6201 sues the NLME-NETWORK-FORMATION.confirm primitive with the Status parameter set to
6202 STARTUP_FAILURE.

6203 If only a single channel is provided in the higher layer request, the NLME does not need to request an en-
6204 ergy scan prior to starting the network. An active scan is still conducted to ensure a PAN identifier is se-
6205 lected that does not conflict with that of any network known to be operating.

6206 Next, the NLME issues the MLME-START.request primitive to the MAC sub-layer. The PANCoordinator
6207 parameter of the MLME-START.request primitive is set to TRUE. The BeaconOrder, SuperframeOrder,
6208 and BatteryLifeExtension parameters will have the same values as those given to the
6209 NLME-NETWORK-FORMATION.request. The CoordRealignment parameter in the
6210 MLME-START.request primitive is set to FALSE if the primitive is issued to start a new PAN. The Coor-
6211 dRealignment parameter is set to TRUE if the primitive is issued to change any of the PAN configuration
6212 attributes on an existing PAN. On receipt of the associated MLME-START.confirm primitive, the NLME
6213 issues the NLME-NETWORK-FORMATION.confirm primitive to the next higher layer with the status re-
6214 turned from the MLME-START.confirm primitive.

6215 **3.2.2.4 NLME-NETWORK-FORMATION.confirm**

6216 This primitive reports the results of the request to initialize a ZigBee coordinator in a network.

6217 **3.2.2.4.1 Semantics of the Service Primitive**

6218 The semantics of this primitive are as follows:

```
6219 NLME-NETWORK-FORMATION.confirm      {  
6220     Status  
6221 }
```

6222 Table 3.10 specifies the parameters for the NLME-NETWORK-FORMATION.confirm primitive.

6223 **Table 3.10 NLME-NETWORK-FORMATION.confirm Parameters**

Name	Type	Valid Range	Description
Status	Status	INVALID_REQUEST, STARTUP_FAILURE or any status value returned from the MLME-START.confirm primitive	The result of the attempt to initialize a ZigBee coordinator.

6225 **3.2.2.4.2 When Generated**

6226 This primitive is generated by the NLME and issued to its next higher layer in response to an
6227 NLME-NETWORK-FORMATION.request primitive. This primitive returns a status value of INVA-
6228 LID_REQUEST, STARTUP_FAILURE or any status value returned from the MLME-START.confirm
6229 primitive. Conditions under which these values may be returned are described in section 3.2.2.3.3.

6230 **3.2.2.4.3 Effect on Receipt**

6231 On receipt of this primitive, the next higher layer is notified of the results of its request to initialize the
6232 device as a ZigBee coordinator. If the NLME has been successful, the Status parameter will be set to SUC-
6233 CESS. Otherwise, the Status parameter indicates the error.

6234 **3.2.2.5 NLME-PERMIT-JOINING.request**

6235 This primitive allows the next higher layer of a ZigBee coordinator or router to set its MAC sub-layer asso-
6236 ciation permit flag for a fixed period when it may accept devices onto its network.

6237 **3.2.2.5.1 Semantics of the Service Primitive**

6238 The semantics of this primitive are as follows:

```
6239 NLME-PERMIT-JOINING.request      {
```

6240 PermitDuration
6241 }
6242

6243 Table 3.11 specifies the parameters for the NLME-PERMIT-JOINING.request primitive.
6244

Table 3.11 NLME-PERMIT-JOINING.request Parameters

Name	Type	Valid Range	Description
PermitDuration	Integer	0x00 – 0xff	The length of time in seconds during which the ZigBee coordinator or router will allow associations. The value 0x00 and 0xff indicate that permission is disabled or enabled, respectively, without a specified time limit.

6245 **3.2.2.5.2 When Generated**

6246 This primitive is generated by the next higher layer of a ZigBee coordinator or router and issued to its
6247 NLME whenever it would like to permit or prohibit the joining of the network by new devices.

6248 **3.2.2.5.3 Effect on Receipt**

6249 It is only permissible that the next higher layer of a ZigBee coordinator or router issue this primitive. On
6250 receipt of this primitive by the NWK layer of a ZigBee end device, the NLME-PERMIT-JOINING.confirm
6251 primitive returns a status of INVALID_REQUEST.

6252 On receipt of this primitive with the PermitDuration parameter set to 0x00, the NLME sets the MAC PIB
6253 attribute, *macAssociationPermit*, to FALSE by issuing the MLME-SET.request primitive to the MAC
6254 sub-layer. Once the MLME-SET.confirm primitive is received, the NLME issues the
6255 NLME-PERMIT-JOINING.confirm primitive with a status equal to that received from the MAC sub-layer.

6256 On receipt of this primitive with the PermitDuration parameter set to 0xff, the NLME sets the MAC PIB
6257 attribute, *macAssociationPermit*, to TRUE by issuing the MLME-SET.request primitive to the MAC
6258 sub-layer. Once the MLME-SET.confirm primitive is received, the NLME issues the
6259 NLME-PERMIT-JOINING.confirm primitive with a status equal to that received from the MAC sub-layer.

6260 On receipt of this primitive with the PermitDuration parameter set to any value other than 0x00 or 0xff, the
6261 NLME sets the MAC PIB attribute, *macAssociationPermit*, to TRUE as described above. Following the
6262 receipt of the MLME-SET.confirm primitive, the NLME starts a timer to expire after PermitDuration
6263 seconds. Once the timer is set, the NLME issues the NLME-PERMIT-JOINING.confirm primitive with a
6264 status equal to that received by the MAC sub-layer. On expiration of the timer, the NLME sets *macAssociationPermit*
6265 to FALSE by issuing the MLME-SET.request primitive.

6266 Every NLME-PERMIT-JOINING.request primitive issued by the next higher layer supersedes all previous
6267 requests.

6268 **3.2.2.6 NLME-PERMIT-JOINING.confirm**

6269 This primitive allows the next higher layer of a ZigBee coordinator or router to be notified of the results of
6270 its request to permit the acceptance of devices onto the network.

6271 **3.2.2.6.1 Semantics of the Service Primitive**

6272 The semantics of this primitive are as follows:

NLME-PERMIT-JOINING.confirm	{
	Status

6275 }

6276

6277 Table 3.12 specifies the parameters for the NLME-PERMIT-JOINING.confirm primitive.

6278 **Table 3.12 NLME-PERMIT-JOINING.confirm Parameters**

Name	Type	Valid Range	Description
Status	Status	INVALID_REQUEST or any status returned from the MLME-SET.confirm primitive (see [B1]).	The status of the corresponding request

6279 **3.2.2.6.2 When Generated**

6280 This primitive is generated by the initiating NLME of a ZigBee coordinator or router and issued to its next
6281 higher layer in response to an NLME-PERMIT-JOINING.request. The Status parameter either indicates the
6282 status received from the MAC sub-layer or an error code of INVALID_REQUEST. The reasons for these
6283 status values are described in section 3.2.2.5.

6284 **3.2.2.6.3 Effect on Receipt**

6285 On receipt of this primitive, the next higher layer of the initiating device is notified of the results of its re-
6286 quest to permit devices to join the network.

6287 **3.2.2.7 NLME-START-ROUTER.request**

6288 This primitive allows the next higher layer of a ZigBee router to initiate the activities expected of a ZigBee
6289 router including the routing of data frames, route discovery, and the accepting of requests to join the net-
6290 work from other devices.

6291 **3.2.2.7.1 Semantics of the Service Primitive**

6292 The semantics of this primitive are as follows:

```
6293 NLME-START-ROUTER.request {
6294     BeaconOrder,
6295     SuperframeOrder,
6296     BatteryLifeExtension
6297 }
```

6298

6299 Table 3.13 specifies the parameters for NLME-START-ROUTER.request.

6300 **Table 3.13 NLME-START-ROUTER.request Parameters**

Name	Type	Valid Range	Description
BeaconOrder	Integer	0x00 – 0x0f	The beacon order of the network that the higher layers wish to form.

SuperframeOrder	Integer	0x00 – 0x0f	The superframe order of the network that the higher layers wish to form.
BatteryLifeExtension	Boolean	TRUE or FALSE	If this value is TRUE, the NLME will request that the ZigBee router is started supporting battery life extension mode; If this value is FALSE, the NLME will request that the ZigBee router is started without supporting battery life extension mode.

6301 **3.2.2.7.2 When Generated**

6302 This primitive is generated by the next higher layer of a new device and issued to its NLME to request the
6303 initialization of itself as a ZigBee router.

6304 **3.2.2.7.3 Effect on Receipt**

6305 On receipt of this primitive by a device that is not already joined to a ZigBee network as a router, the
6306 NLME issues the NLME-START-ROUTER.confirm primitive with the Status parameter set to INVA-
6307 LID_REQUEST.

6308 On receipt of this primitive by the NLME of a device that is joined to a ZigBee network as a router, the
6309 NLME issues the MLME-START.request primitive to the MAC sub-layer. The BeaconOrder, Super-
6310 frameOrder, and BatteryLifeExtension parameters of the MLME-START.request primitive will have the
6311 values given by the corresponding parameters of the NLME-START-ROUTER.request. The CoordRe-
6312 alignment parameter in the MLME-START.request primitive is set to FALSE if the primitive is issued to
6313 start as a router for the first time. The CoordRealignment parameter is set to TRUE thereafter if the primi-
6314 tive is issued to change any of the PAN configuration attributes.

6315 On receipt of the associated MLME-START.confirm primitive, the NLME issues the
6316 NLME-START-ROUTER.confirm primitive to the next higher layer with the status returned from the
6317 MLME-START.confirm primitive. If, and only if, the status returned from the MLME-START.confirm
6318 primitive is SUCCESS, the device may then begin to engage in the activities expected of a ZigBee router
6319 including the routing of data frames, route discovery, and the accepting of requests to join the network from
6320 other devices. Otherwise, the device is expressly forbidden to engage in these activities.

6321 **3.2.2.8 NLME-START-ROUTER.confirm**

6322 This primitive reports the results of the request to initialize a ZigBee router.

6323 **3.2.2.8.1 Semantics of the Service Primitive**

6324 The semantics of this primitive are as follows:

```
NLME-START-ROUTER.confirm {
    Status
}
```

6328

6329 Table 3.14 specifies the parameters for NLME-START-ROUTER.confirm.

6330

Table 3.14 NLME-START-ROUTER.confirm Parameters

Name	Type	Valid Range	Description
Status	Status	INVALID_REQUEST or any status value returned from the MLME-START.confirm primitive.	The result of the attempt to initialize a ZigBee router.

6331

3.2.2.8.2 When Generated

This primitive is generated by the NLME and issued to its next higher layer in response to an NLME-START-ROUTER.request primitive. This primitive returns a status value of INVALID_REQUEST or any status value returned from the MLME-START.confirm primitive. Conditions under which these values may be returned are described in section 3.2.2.7.3.

3.2.2.8.3 Effect on Receipt

On receipt of this primitive, the next higher layer is notified of the results of its request to initialize a ZigBee router. If the NLME has been successful, the Status parameter will be set to SUCCESS. Otherwise, the Status parameter indicates the error.

3.2.2.9 NLME-ED-SCAN.request

This primitive allows the next higher layer to request an energy scan to evaluate channels in the local area.

3.2.2.9.1 Semantics of the Service Primitive

The semantics of this primitive are as follows:

NLME-ED-SCAN.request	{
	ScanChannels,
	ScanDuration,
	}

Table 3.15 specifies the parameters for the service primitive.

Table 3.15 NLME-ED-SCAN.request Parameters

Name	Type	Valid Range	Description
ScanChannels	Bitmap	32-bit field	The five most significant bits (b27,..., b31) are reserved. The 27 least significant bits (b0, b1,... b26) indicate which channels are to be scanned (1=scan, 0=do not scan) for each of the 27 valid channels (see [B1]).

ScanDuration	Integer	0x00-0x0e	A value used to calculate the length of time to spend scanning each channel. The time spent scanning each channel is ($aBaseSuperframeDuration * (2^n + 1)$) symbols, where n is the value of the ScanDuration parameter [B1].
--------------	---------	-----------	--

3.2.2.9.2 When Generated

The next higher layer of a device generates this primitive to request to conduct an energy scan of channels.

3.2.2.9.3 Effect on Receipt

On receipt of this primitive by a device that is currently joined to a network, the device will temporarily stop operating on the network to conduct an energy scan. To do this, the NLME issues the MLME-SCAN.request primitive to the MAC sub-layer with the ScanType parameter set to indicate an energy detection scan and the ScanChannels and ScanDuration from the NLME request and the ChannelPage set to zero.

3.2.2.10 NLME-ED-SCAN.confirm

This primitive provides the next higher layer results from an energy scan.

3.2.2.10.1 Semantics of the Service Primitive

The semantics of this primitive are as follows:

NLME-ED-SCAN.confirm	{
	Status
	EnergyDetectList
	}

Table 3.16 specifies the parameters for the service primitive.

Table 3.16 NLME-ED-SCAN.confirm

Name	Type	Valid Range	Description
Status	Status	SUCCESS, or any valid code from the MAC	The status of the request.
EnergyDetectList	List of integers	0x00-0xff for each integer	The list of energy measurements in accordance with [B1], one for each channel.

3.2.2.10.2 When Generated

This primitive is generated by the NLME of a ZigBee device in response to an NLME-ED-SCAN.request. The status indicates the status received from the MAC sub-layer primitive MLME-SCAN.confirm. EnergyDetectList contains the ED scan result (List of integers, 0x00 - 0xff) for the channels scanned in accordance with IEEE 802.15.4-2003.

3.2.2.10.3 Effect on Receipt

On receipt of this primitive, the next higher layer is notified of the results of an ED scan.

6378 **3.2.2.11 NLME-JOIN.request**
6379 This primitive allows the next higher layer to request to join or rejoin a network, or to change the operating
6380 channel for the device while within an operating network.

6381 **3.2.2.11.1 Semantics of the Service Primitive**

6382 The semantics of this primitive are as follows:

6383 NLME-JOIN.request {
6384 ExtendedPANId,
6385 RejoinNetwork,
6386 ScanChannels,
6387 ScanDuration,
6388 CapabilityInformation,
6389 SecurityEnable
6390 }

6391
6392 Table 3.17 specifies the parameters for the NLME-JOIN.request primitive.

Table 3.17 NLME-JOIN.request

Name	Type	Valid Range	Description
ExtendedPANId	Integer	0x0000000000000001 – 0xffffffffffffffe	The 64-bit PAN identifier of the network to join.
RejoinNetwork	Integer	0x00 – 0x03	<p>This parameter controls the method of joining the network.</p> <p>The parameter is 0x00 if the device is requesting to join a network through association.</p> <p>The parameter is 0x01 if the device is joining directly or rejoining the network using the orphaning procedure.</p> <p>The parameter is 0x02 if the device is joining the network using the NWK rejoicing procedure.</p> <p>The parameter is 0x03 if the device is to change the operational network channel to that identified in the ScanChannels parameter.</p>
ScanChannels	Bitmap	32-bit field	The five most significant bits (b27,..., b31) are reserved. The 27 least significant bits (b0, b1,... b26) indicate which channels are to be scanned (1=scan, 0=do not scan) for each of the 27 valid channels (see [B1]).
ScanDuration	Integer	0x00-0x0e	A value used to calculate the length of time to spend scanning each channel. The time spent scanning each channel is ($aBaseSuperframe-Duration * (2^n + 1)$) symbols, where n is the value of the ScanDuration parameter [B1].
CapabilityInformation	Bitmap	See Table 3.52.	The operating capabilities of the device being directly joined.
SecurityEnable	Boolean	TRUE or FALSE	If the value of RejoinNetwork is 0x02 and this is TRUE than the device will try to rejoin securely. Otherwise, this is set to FALSE.

6394 **3.2.2.11.2 When Generated**

6395 The next higher layer of a device generates this primitive to request to:

- 6396 • Join a network using the MAC association procedure.
6397 • Join or rejoin a network using the orphaning procedure.
6398 • Join or rejoin a network using the NWK rejoin procedure.
6399 • Switch the operating channel for a device that is joined to a network.

6400 **3.2.2.11.3 Effect on Receipt**

6401 On receipt of this primitive by a device that is currently joined to a network and with the RejoinNetwork
6402 parameter equal to 0x00, the NLME issues an NLME-JOIN.confirm primitive with the Status parameter set
6403 to INVALID_REQUEST.

6404 On receipt of this primitive by a device that is not currently joined to a network and with the RejoinNet-
6405 work parameter equal to 0x00, the device attempts to join the network specified by the 64-bit Extended-
6406 PANId parameter as described in section 3.6.1.4.1.1.

6407 Whether joining or rejoining, the device shall set the nwkParentInformation in the NIB to 0.

6408 If a device receives this primitive and the RejoinNetwork parameter is equal to 0x01, then it attempts to
6409 join or rejoin the network using orphaning as described in section 3.6.1.4.3.2.

6410 On receipt of this primitive with the RejoinNetwork parameter is equal to 0x02, the device attempts to re-
6411 join the network with 64-bit extended PAN ID given by the ExtendedPANId parameter. The procedure for
6412 rejoining is given in section 3.6.1.4.2.

6413 Once the device has successfully joined a network, it will set the value of the *nwkExtendedPANId* NIB at-
6414 tribute to the extended PAN identifier of the network to which it joined.

6415 If a device receives this primitive and the RejoinNetwork parameter is equal to 0x03, and the device sup-
6416 ports setting the value of phyCurrentChannel then the device attempts to switch the operating channel to
6417 that provided in the ScanChannels parameter. If more than one channel is provided in the ScanChannels
6418 parameter, the NLME issues an NLME-JOIN.confirm primitive with the Status parameter set to INVA-
6419 LID_REQUEST. If the channel change is performed successfully, then the device issues a
6420 NLME-JOIN.confirm with the Status parameter set to SUCCESS. If the device does not support the setting
6421 of phyCurrentChannel directly, then the NLME issues a NLME-JOIN.confirm primitive with the Status
6422 parameter set to UNSUPPORTED_ATTRIBUTE.

6423 If the MAC layer returned an error status during the channel change then this status shall be reported in the
6424 status field of the NLME-JOIN.confirm primitive.

6425 **3.2.2.12 NLME-JOIN.indication**

6426 This primitive allows the next higher layer of a ZigBee coordinator or ZigBee router to be notified when a
6427 new device has successfully joined its network by association or rejoined using the NWK rejoin procedure
6428 as described in section 3.6.1.4.3.

6429 **3.2.2.12.1 Semantics of the Service Primitive**

6430 The semantics of this primitive are as follows:

6431 NLME-JOIN.indication	{
6432	NetworkAddress,
6433	ExtendedAddress,
6434	CapabilityInformation,
6435	RejoinNetwork
6436	SecureRejoin

6437 }

6438

6439 Table 3.18 specifies the parameters for the NLME-JOIN.indication primitive.

6440 **Table 3.18 NLME-JOIN.indication Parameters**

Name	Type	Valid Range	Description
NetworkAddress	Network address	0x0001 – 0xffff	The network address of an entity that has been added to the network.
ExtendedAddress	64-bit IEEE address	Any 64-bit, IEEE address	The 64-bit IEEE address of an entity that has been added to the network.
CapabilityInformation	Bitmap	See [B1]	Specifies the operational capabilities of the joining device.
RejoinNetwork	Integer	0x00 - 0x02	The RejoinNetwork parameter indicating the method used to join the network. The parameter is 0x00 if the device joined through association. The parameter is 0x01 if the device joined directly or rejoined using orphaning. The parameter is 0x02 if the device used NWK rejoin.
SecureRejoin	Boolean	TRUE or FALSE	This parameter will be TRUE if the rejoin was performed in a secure manner. Otherwise, this parameter will be FALSE.

6441 **3.2.2.12.2 When Generated**

6442 This primitive is generated by the NLME of a ZigBee coordinator or router and issued to its next higher
6443 layer on successfully adding a new device to the network using the MAC association procedure as shown in
6444 Figure 3.37, or on allowing a device to rejoin the network using the NWK rejoicing procedure as shown in
6445 Figure 3.42.

6446 **3.2.2.12.3 Effect on Receipt**

6447 On receipt of this primitive, the next higher layer of a ZigBee coordinator or ZigBee router is notified that a
6448 new device has joined its network.

6449 **3.2.2.13 NLME-JOIN.confirm**

6450 This primitive allows the next higher layer to be notified of the results of its request to join a network.

6451 **3.2.2.13.1 Semantics of the Service Primitive**

6452 The semantics of this primitive are as follows:

6453 NLME-JOIN.confirm {
6454 Status,
6455 NetworkAddress,
6456 ExtendedPANID,
6457 ActiveChannel
6458 }

6459
6460 Table 3.19 specifies the parameters for the NLME-JOIN.confirm primitive.

6461 **Table 3.19 NLME-JOIN.confirm**

Name	Type	Valid Range	Description
Status	Status	INVALID_REQUEST, NOT_PERMITTED, NO_NETWORKS or any status value returned from the MLME-ASSOCIATE.confirm primitive, the MLME-SCAN.confirm primi- tive or the PLME-SET.confirm	The status of the corresponding request.
NetworkAddress	Integer	0x0001 – 0xffff and 0xffff	The 16-bit network address that was al- located to this device. This parameter will be equal to 0xffff if the join attempt was unsuccessful.
ExtendedPANID	Integer	0x0000000000000001 – 0xfffffffffffffe	The 64-bit extended PAN identifier for the network of which the device is now a member.
ActiveChannel	Integer	The number of any channel supported by the PHY (see [B1]).	The value of <i>phyCurrentChannel</i> attrib- ute of the PHY PIB, which is equal to the current channel of the network that has been joined.

6462 **3.2.2.13.2 When Generated**

6463 This primitive is generated by the initiating NLME and issued to its next higher layer in response to an
6464 NLME-JOIN.request primitive. If the request was successful, the Status parameter will have a value of
6465 SUCCESS. Otherwise, the Status parameter indicates an error code of INVALID_REQUEST,
6466 NOT_PERMITTED, NO_NETWORKS or any status value returned from either the
6467 MLME-ASSOCIATE.confirm primitive, the MLME-SCAN.confirm primitive or the PLME-SET.confirm
6468 primitive. The reasons for these status values are fully described in section 3.2.2.11.3.

6469 **3.2.2.13.3 Effect on Receipt**
6470 On receipt of this primitive, the next higher layer of the initiating device is notified of the results of its re-
6471 quest to join a network using the MAC sub-layer association procedure, to join directly using the MAC
6472 sub-layer orphaning procedure, or to re-join a network once it has been orphaned.

6473 **3.2.2.14 NLME-DIRECT-JOIN.request**

6474 This optional primitive allows the next higher layer of a ZigBee coordinator or router to request to directly
6475 join another device to its network.

6476 **3.2.2.14.1 Semantics of the Service Primitive**

6477 The semantics of this optional primitive are as follows:

6478 NLME-DIRECT-JOIN.request {
6479 DeviceAddress,
6480 CapabilityInformation
6481 }

6482
6483 Table 3.20 specifies the parameters for the NLME-DIRECT-JOIN.request primitive.
6484

Table 3.20 NLME-DIRECT-JOIN.request Parameters

Name	Type	Valid Range	Description
DeviceAddress	64-bit IEEE address	Any 64-bit IEEE address	The IEEE address of the device to be directly joined.
CapabilityInformation	Bitmap	See Table 3.52.	The operating capabilities of the device being directly joined.

6485 **3.2.2.14.2 When Generated**

6486 The next higher layer of a ZigBee coordinator or router generates this primitive to add a new device directly
6487 to its network. This process is completed without any over the air transmissions.

6488 **3.2.2.14.3 Effect on Receipt**

6489 On receipt of this primitive, the NLME will attempt to add the device specified by the DeviceAddress parameter
6490 to its neighbor table. The CapabilityInformation parameter will contain a description of the device
6491 being joined. The alternate PAN coordinator bit is set to 0 in devices implementing this specification. The
6492 device type bit is set to 1 if the device is a ZigBee router, or to 0 if it is an end device. The power source bit
6493 is set to 1 if the device is receiving power from the alternating current mains or to 0 otherwise. The receiver
6494 on when idle bit is set to 1 if the device does not disable its receiver during idle periods, or to 0 otherwise.
6495 The security capability bit is set to 1 if the device is capable of secure operation, or to 0 otherwise.

6496 If the NLME successfully adds the device to its neighbor table, the NLME issues the
6497 NLME-DIRECT-JOIN.confirm primitive with a status of SUCCESS. If the NLME finds that the requested
6498 device is already present in its neighbor tables, the NLME issues the NLME-DIRECT-JOIN.confirm primitive
6499 with a status of ALREADY_PRESENT. If no capacity is available to add a new device to the device
6500 list, the NLME issues the NLME-DIRECT-JOIN.confirm primitive with a status of NEIGHBOR_TABLE_FULL.
6501

6502 **3.2.2.15 NLME-DIRECT-JOIN.confirm**

6503 This primitive allows the next higher layer of a ZigBee coordinator or router to be notified of the results of
6504 its request to directly join another device to its network.

6505 **3.2.2.15.1 Semantics of the Service Primitive**

6506 The semantics of this primitive are as follows:

6507 NLME-DIRECT-JOIN.confirm {
6508 Status,
6509 DeviceAddress
6510 }

6511 Table 3.21 specifies the parameters for the NLME-DIRECT-JOIN.confirm primitive.

6513 **Table 3.21 NLME-DIRECT-JOIN.confirm Parameters**

Name	Type	Valid Range	Description
Status	Status	SUCCESS, ALREADY_PRESENT, NEIGHBOR_TABLE_FULL	The status of the corresponding request.
DeviceAddress	64-bit IEEE address	Any 64-bit, IEEE address	The 64-bit IEEE address in the request to which this is a confirmation.

6514 **3.2.2.15.2 When Generated**

6515 This primitive is generated by the initiating NLME and issued to its next higher layer in response to an
6516 NLME-DIRECT-JOIN.request primitive. If the request was successful, the Status parameter indicates a
6517 successful join attempt. Otherwise, the Status parameter indicates an error code of ALREADY_PRESENT
6518 or NEIGHBOR_TABLE_FULL. The reasons for these status values are fully described in section
6519 3.2.2.14.3.

6520 **3.2.2.15.3 Effect on Receipt**

6521 On receipt of this primitive, the next higher layer of the initiating device is notified of the results of its re-
6522 quest to directly join another device to a network.

6523 **3.2.2.16 NLME-LEAVE.request**

6524 This primitive allows the next higher layer to request that it or another device leaves the network.

6525 **3.2.2.16.1 Semantics of the Service Primitive**

6526 The semantics of this primitive are as follows:

6527 NLME-LEAVE.request {
6528 DeviceAddress,
6529 RemoveChildren,
6530 Rejoin
6531 }

6532

6533 Table 3.22 specifies the parameters for the NLME-LEAVE.request primitive.

6534 **Table 3.22 NLME-LEAVE.request Parameters**

Name	Type	Valid Range	Description
DeviceAddress	Device address	Any 64-bit, IEEE address	The 64-bit IEEE address of the entity to be removed from the network or NULL if the device removes itself from the network.
RemoveChildren	Boolean	TRUE or FALSE	This parameter has a value of TRUE if the device being asked to leave the network is also being asked to remove its child devices, if any. Otherwise, it has a value of FALSE.
Rejoin	Boolean	TRUE or FALSE	This parameter has a value of TRUE if the device being asked to leave from the current parent is requested to rejoin the network. Otherwise, the parameter has a value of FALSE. Note that the Rejoin parameter is set by the application so cannot be relied upon by the networking layer to indicate whether a Join or Rejoin request will be accepted in the future.

6535

3.2.2.16.2 When Generated

6536
6537
6538

The next higher layer of a device generates this primitive to request to leave the network. The next higher layer of a ZigBee coordinator or router may also generate this primitive to remove a device from the network.

6539

3.2.2.16.3 Effect on Receipt

6540
6541
6542
6543
6544
6545
6546
6547
6548
6549
6550
6551

On receipt of this primitive by the NLME of a device that is not currently joined to a network, the NLME issues the NLME-LEAVE.confirm primitive with a status of INVALID_REQUEST. On receipt of this primitive by the NLME of a device that is currently joined to a network, with the DeviceAddress parameter equal to the local device's IEEE address or NULL, the NLME will remove itself from the network using the procedure described in section 3.6.1.10.1, and the value of the Rejoin parameter shall be copied into the Network Leave command frame that is generated. If the Rejoin parameter is set to TRUE, no further action is taken. If the Rejoin parameter is set to FALSE the NLME will then clear its routing table and route discovery table and issue an MLME-RESET.request primitive to the MAC sub-layer. The NLME will also set the relationship field of the neighbor table entry corresponding to its former parent to 0x03, indicating no relationship. If the NLME-LEAVE.request primitive is received with the DeviceAddress parameter equal to NULL and the RemoveChildren parameter equal to TRUE, then the NLME will attempt to remove the device's children, as described in section 3.6.1.10.3.

6552 On receipt of this primitive by a ZigBee coordinator or ZigBee router and with the DeviceAddress parameter not equal to NULL and not equal to the local device's IEEE address, the NLME determines whether the specified device is in the Neighbor Table and the device type is 0x02 (Zigbee End device). If the requested device does not exist or the device type is not 0x02, the NLME issues the NLME-LEAVE.confirm primitive with a status of UNKNOWN_DEVICE. If the requested device exists, the NLME will attempt to remove it from the network using the procedure described in section 3.6.1.10.3. If the RemoveChildren parameter is equal to TRUE then the device will be requested to remove its children as well. Following the removal, the NLME will issue the NLME-LEAVE.confirm primitive with the DeviceAddress equal to the 64-bit IEEE address of the removed device and the Status parameter equal to the status delivered by the MCPS-DATA.confirm primitive. If the relationship field of the neighbor table entry corresponding to the leaving device has a value of 0x01 then it will be changed to 0x04 indicating previous child. If it has a value of 0x05, indicating that the child has not yet authenticated, it will be changed to 0x03, indicating no relationship.

6565 **3.2.2.17 NLME-LEAVE.indication**

6566 This primitive allows the next higher layer of a ZigBee device to be notified if that ZigBee device has left
6567 the network or if a neighboring device has left the network.

6568 **3.2.2.17.1 Semantics of the Service Primitive**

6569 The semantics of this primitive are as follows:

6570 NLME-LEAVE.indication {
6571 DeviceAddress,
6572 Rejoin
6573 }

6574
6575 Table 3.23 specifies the parameters for the NLME-LEAVE.indication primitive.
6576

Table 3.23 NLME-LEAVE.indication Parameters

Name	Type	Valid Range	Description
DeviceAddress	64-bit IEEE address	Any 64-bit, IEEE address	The 64-bit IEEE address of an entity that has removed itself from the network or NULL in the case that the device issuing the primitive has been removed from the network by its parent.
Rejoin	Boolean	TRUE or FALSE	This parameter has a value of TRUE if the device being asked to leave the current parent is requested to rejoin the network. Otherwise, this parameter has a value of FALSE.

6577 **3.2.2.17.2 When Generated**

6578 This primitive is generated by the NLME of a ZigBee coordinator or ZigBee router and issued to its next
6579 higher layer on receipt of a broadcast leave command pertaining to a device on its PAN. It is also generated
6580 by the NLME of a ZigBee router or end device and issued to its next higher layer to indicate that it has
6581 been successfully removed from the network by its associated router or ZigBee coordinator.

6582 **3.2.2.17.3 Effect on Receipt**

6583 On receipt of this primitive, the next higher layer of a ZigBee coordinator or ZigBee router is notified that a
6584 device, formerly on the network, has left. The primitive can also inform the next higher layer of a ZigBee
6585 router or end device that it has been removed from the network by its associated ZigBee router or ZigBee
6586 coordinator parent. In this case, the value of the Rejoin parameter indicates to the next higher layer whether
6587 the peer entity on the parent device wishes the device that has been removed to rejoin the same network.

6588 When the local device receives a NLME-LEAVE.indication with Rejoin set to FALSE it shall remove any
6589 persistent data within the stack related to the leaving device.

6590 **3.2.2.18 NLME-LEAVE.confirm**

6591 This primitive allows the next higher layer of the initiating device to be notified of the results of its request
6592 for itself or another device to leave the network.

6593 **3.2.2.18.1 Semantics of the Service Primitive**

6594 The semantics of this primitive are as follows:

6595 NLME-LEAVE.confirm {
6596 Status,
6597 DeviceAddress
6598 }

6599
6600 Table 3.24 specifies the parameters for the NLME-LEAVE.confirm primitive.
6601

Table 3.24 NLME-LEAVE.confirm Parameters

Name	Type	Valid Range	Description
Status	Status	SUCCESS, INVALID_REQUEST, UNKNOWN_DEVICE or any status returned by the MCPS-DATA.confirm primitive	The status of the corresponding request.
DeviceAddress	64-bit IEEE address	Any 64-bit, IEEE address	The 64-bit IEEE address in the request to which this is a confirmation or null if the device requested to remove itself from the network.

6602 **3.2.2.18.2 When Generated**

6603 This primitive is generated by the initiating NLME and issued to its next higher layer in response to an
6604 NLME-LEAVE.request primitive. If the request was successful, the Status parameter indicates a successful
6605 leave attempt. Otherwise, the Status parameter indicates an error code of INVALID_REQUEST, UN-
6606 KNOWN_DEVICE or a status delivered by the MCPS-DATA.confirm primitive. The reasons for these
6607 status values are fully described in section 3.2.2.16.3.

6608 **3.2.2.18.3 Effect on Receipt**

6609 On receipt of this primitive, the next higher layer of the initiating device is notified of the results of its re-
6610 quest for itself or a child device to leave the network.

6611 **3.2.2.19 NLME-RESET.request**

6612 This primitive allows the next higher layer to request the NWK layer to perform a reset operation.

6613 **3.2.2.19.1 Semantics of the Service Primitive**

6614 The semantics of this primitive are as follows:

6615 NLME-RESET.request {
6616 WarmStart
6617 }

6618 Table 3.25 specifies the parameters for this primitive.

6619 **Table 3.25 NLME-RESET.request Parameters**

Name	Type	Valid Range	Description
WarmStart	Boolean	TRUE or FALSE	This parameter has a value of FALSE if the request is expected reset all stack values to their initial default values. If this value is TRUE, the device is expected to resume operations according to the NIB settings prior to the call.

6621 **3.2.2.19.2 When Generated**

6622 This primitive is generated by the next higher layer and issued to its NLME to request the NWK layer be
6623 reset to its initial condition, or that it resume operations according to its current NIB values prior to this
6624 primitive being issued.

6625 **3.2.2.19.3 Effect on Receipt**

6626 On receipt of this primitive, where the WarmStart parameter has a value of FALSE, the NLME issues the
6627 MLME-RESET.request primitive to the MAC sub-layer with the SetDefaultPIB parameter set to TRUE.
6628 On receipt of the corresponding MLME-RESET.confirm primitive, the NWK layer resets itself by clearing
6629 all internal variables, routing table and route discovery table entries and by setting all NIB attributes to their
6630 default values. Once the NWK layer is reset, the NLME issues the NLME-RESET.confirm with the Status
6631 parameter set to SUCCESS if the MAC sub-layer was successfully reset or DISABLE_TRX_FAILURE
6632 otherwise.

6633 On receipt of this primitive where WarmStart is set to TRUE, the network layer should not modify any NIB
6634 values, but rather should resume normal network operations and consider itself part of the network specified
6635 in the NIB. Routing table values and neighbor table values should be cleared. The method by which
6636 the network and MAC layers attributes are pre-loaded is left to the implementer.

6637 If this primitive is issued to the NLME of a device that is currently joined to a network, any required leave
6638 attempts using the NLME-LEAVE.request primitive should be made *a priori* at the discretion of the next
6639 higher layer.

6640 **3.2.2.20 NLME-RESET.confirm**

6641 This primitive allows the next higher layer of the initiating device to be notified of the results of the request
6642 to reset the NWK layer.

6643 **3.2.2.20.1 Semantics of the Service Primitive**

6644 The semantics of this primitive are as follows:

6645 NLME-RESET.confirm {
6646 Status
6647 }

6648

6649 Table 3.26 specifies the parameters for this primitive.

6650 **Table 3.26 NLME-RESET.confirm Parameters**

Name	Type	Valid Range	Description
Status	Status	Any status value returned from the MLME-RESET.confirm primitive (see [B1])	The result of the reset operation

6651 **3.2.2.20.2 When Generated**

6652 This primitive is generated by the NLME and issued to its next higher layer in response to an
6653 NLME-RESET.request primitive. If the request was successful, the Status parameter will have a value of
6654 SUCCESS. Otherwise, the Status parameter will indicate an error code of DISABLE_TRX_FAILURE. The
6655 reasons for these status values are fully described in section 3.2.2.19.3.

6656 **3.2.2.20.3 Effect on Receipt**

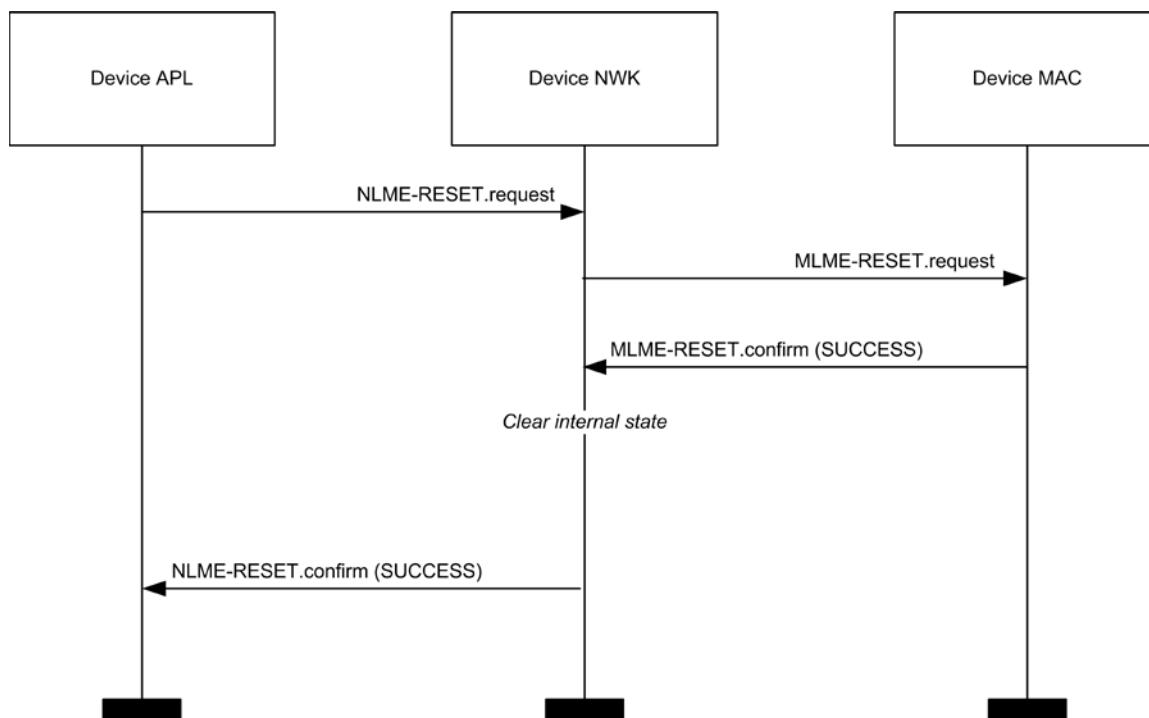
6657 On receipt of this primitive, the next higher layer is notified of the results of its request to reset the NWK
6658 layer.

6659 **3.2.2.21 Network Layer Reset Message Sequence Chart**

6660 Figure 3.2 illustrates the sequence of messages necessary for resetting the NWK layer.

6661

Figure 3.2 Message Sequence Chart for Resetting the Network Layer



6662

3.2.2.22 NLME-SYNC.request

6663 This primitive allows the next higher layer to synchronize or extract data from its ZigBee coordinator or router.
6664
6665

3.2.2.22.1 Semantics of the Service Primitive

6666 The semantics of this primitive are as follows:

6667
6668 NLME-SYNC.request {
6669 Track
6670 }

6671

6672 Table 3.27 specifies the parameters for this primitive.
6673

Table 3.27 NLME-SYNC.request Parameters

Name	Type	Valid Range	Description
Track	Boolean	TRUE or FALSE	Whether or not the synchronization should be maintained for future beacons.

3.2.2.22.2 When Generated

6674 This primitive is generated whenever the next higher layer wishes to achieve synchronization or check for pending data at its ZigBee coordinator or router.
6675
6676

6677 **3.2.2.22.3 Effect on Receipt**

6678 If the Track parameter is set to FALSE and the device is operating on a non-beacon enabled network, the
6679 NLME issues the MLME-POLL.request primitive to the MAC sub-layer. On receipt of the corresponding
6680 MLME-POLL.confirm primitive, the NLME issues the NLME-SYNC.confirm primitive with the Status
6681 parameter set to the value reported in the MLME-POLL.confirm.

6682 If the Track parameter is set to FALSE and the device is operating on a beacon enabled network, the
6683 NLME first sets the *macAutoRequest* PIB attribute in the MAC sub-layer to TRUE by issuing the
6684 MLME-SET.request primitive. It then issues the MLME-SYNC.request primitive with the TrackBeacon
6685 parameter set to FALSE. The NLME then issues the NLME-SYNC.confirm primitive with the Status pa-
6686 rameter set to SUCCESS.

6687 If the Track parameter is set to TRUE and the device is operating on a non-beacon enabled network, the
6688 NLME will issue the NLME-SYNC.confirm primitive with a Status parameter set to INVA-
6689 LID_PARAMETER.

6690 If the Track parameter is set to TRUE and the device is operating on a beacon enabled network, the NLME
6691 first sets the *macAutoRequest* PIB attribute in the MAC sub-layer to TRUE by issuing the
6692 MLME-SET.request primitive. It then issues the MLME-SYNC.request primitive with the TrackBeacon
6693 parameter set to TRUE. The NLME then issues the NLME-SYNC.confirm primitive with the Status pa-
6694 rameter set to SUCCESS.

6695 **3.2.2.23 NLME-SYNC-LOSS.indication**

6696 This primitive allows the next higher layer to be notified of the loss of synchronization at the MAC
6697 sub-layer.

6698 **3.2.2.23.1 Semantics of the Service Primitive**

6699 The semantics of this primitive are as follows:

6700 NLME-SYNC-LOSS.indication {
6701 }

6702

6703 This primitive has no parameters.

6704 **3.2.2.23.2 When Generated**

6705 This primitive is generated upon receipt of a loss of synchronization notification from the MAC sub-layer
6706 via the MLME-SYNC-LOSS.indication primitive with a LossReason of BEACON_LOST. This follows a
6707 prior NLME-SYNC.request primitive being issued to the NLME.

6708 **3.2.2.23.3 Effect on Receipt**

6709 The next higher layer is notified of the loss of synchronization with the beacon.

6710 **3.2.2.24 NLME-SYNC.confirm**

6711 This primitive allows the next higher layer to be notified of the results of its request to synchronize or ex-
6712 tract data from its ZigBee coordinator or router.

6713 **3.2.2.24.1 Semantics of the Service Primitive**

6714 The semantics of this primitive are as follows:

6715 NLME-SYNC.confirm {
6716 Status
6717 }

6718

6719 Table 3.28 specifies the parameters for this primitive.

6720 **Table 3.28 NLME-SYNC.confirm Parameters**

Name	Type	Valid Range	Description
Status	Status	SUCCESS, SYNC_FAILURE, INVALID_PARAMETER, or any status value returned from the MLME_POLL.confirm primitive (see [B1])	The result of the request to synchronize with the ZigBee coordinator or router.

6721 **3.2.2.24.2 When Generated**

6722 This primitive is generated by the initiating NLME and issued to its next higher layer in response to an
6723 NLME-SYNC.request primitive. If the request was successful, the Status parameter indicates a successful
6724 synchronization attempt. Otherwise, the Status parameter indicates an error code. The reasons for these
6725 status values are fully described in section 3.2.2.22.2.

6726 **3.2.2.24.3 Effect on Receipt**

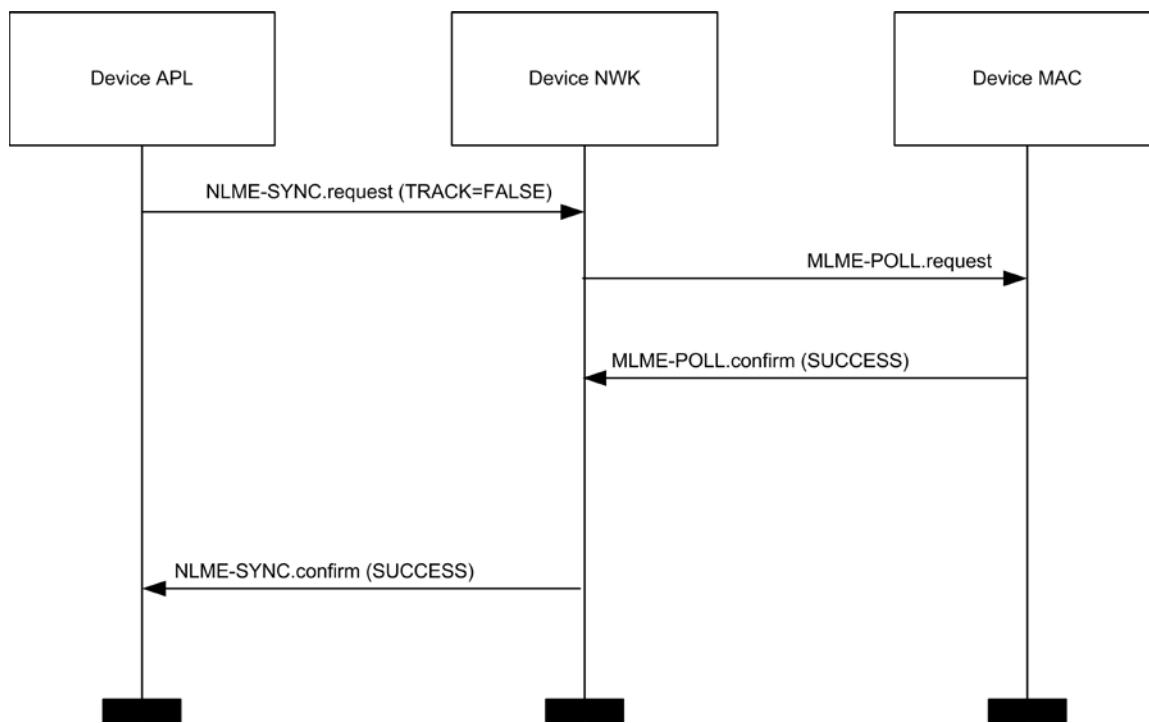
6727 On receipt of this primitive, the next higher layer is notified of the results of its request to synchronize or
6728 extract data from its ZigBee coordinator or router. If the NLME has been successful, the Status parameter
6729 will be set to SUCCESS. Otherwise, the Status parameter indicates the error.

6730 **3.2.2.25 Message Sequence Charts for Synchronization**

6731 Figure 3.3 and Figure 3.4 illustrate the sequence of messages necessary for a device to successfully synchronize with
6732 a ZigBee coordinator or ZigBee router. Figure 3.3 illustrates the case for a non-beaconing network, and Figure 3.4
6733 illustrates the case for a beaconing network.

6734

Figure 3.3 Message Sequence Chart for Synchronizing in a Non-Beaconing Network

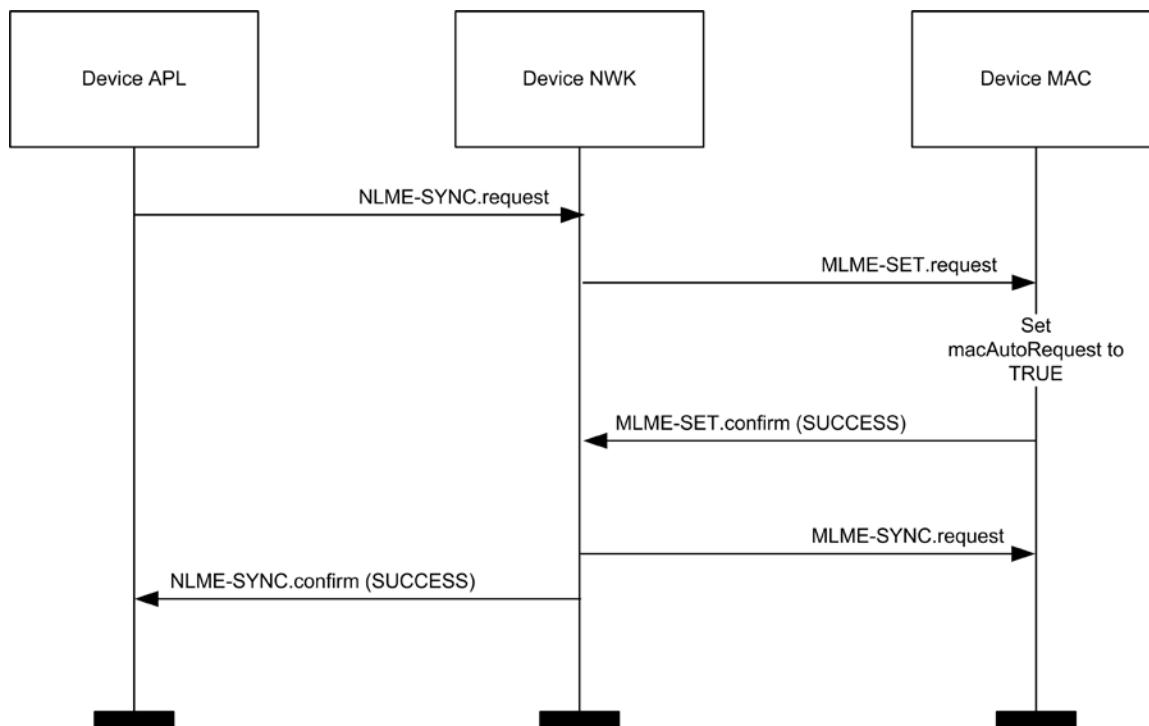


6735

6736

6737

Figure 3.4 Message Sequence Chart for Synchronizing in a Beacon-Enabled Network



6738

6739 **3.2.2.26 NLME-GET.request**

6740 This primitive allows the next higher layer to read the value of an attribute from the NIB.

6741 **3.2.2.26.1 Semantics of the Service Primitive**

6742 The semantics of this primitive are as follows:

6743 NLME-GET.request {
6744 NIBAttribute
6745 }

6746 Table 3.29 specifies the parameters for this primitive.

6748 **Table 3.29 NLME-GET.request Parameters**

Name	Type	Valid Range	Description
NIBAttribute	Integer	See Table 3.49.	The identifier of the NIB attribute to read.

6749 **3.2.2.26.2 When Generated**

6750 This primitive is generated by the next higher layer and issued to its NLME in order to read an attribute
6751 from the NIB.

6752 **3.2.2.26.3 Effect on Receipt**

6753 On receipt of this primitive, the NLME attempts to retrieve the requested NIB attribute from its database. If
6754 the identifier of the NIB attribute is not found in the database, the NLME issues the NLME-GET.confirm
6755 primitive with a status of UNSUPPORTED_ATTRIBUTE.

6756 If the requested NIB attribute is successfully retrieved, the NLME issues the NLME-GET.confirm primitive
6757 with a status of SUCCESS and the NIB attribute identifier and value.

6758 **3.2.2.27 NLME-GET.confirm**

6759 This primitive reports the results of an attempt to read the value of an attribute from the NIB.

6760 **3.2.2.27.1 Semantics of the Service Primitive**

6761 The semantics of this primitive are as follows:

6762 NLME-GET.confirm {
6763 Status,
6764 NIBAttribute,
6765 NIBAttributeLength,
6766 NIBAttributeValue
6767 }

6769 Table 3.30 specifies the parameters for this primitive.

6770 **Table 3.30 NLME-GET.confirm Parameters**

Name	Type	Valid Range	Description
Status	Enumeration	SUCCESS or UNSUPPORTED_ATTRIBUTE	The results of the request to read a NIB attribute value.
NIBAttribute	Integer	See Table 3.49.	The identifier of the NIB attribute that was read.
NIBAttributeLength	Integer	0x0000 – 0xffff	The length, in octets, of the attribute value being returned.
NIBAttributeValue	Various	Attribute specific (see Table 3.49)	The value of the NIB attribute that was read.

6771 **3.2.2.27.2 When Generated**

6772 This primitive is generated by the NLME and issued to its next higher layer in response to an
6773 NLME-GET.request primitive. This primitive returns either a status of SUCCESS, indicating that the re-
6774 quest to read a NIB attribute was successful, or an error code of UNSUPPORTED_ATTRIBUTE. The rea-
6775 sons for these status values are fully described in section 3.2.2.26.3.

6776 **3.2.2.27.3 Effect on Receipt**

6777 On receipt of this primitive, the next higher layer is notified of the results of its request to read a NIB at-
6778 tribute. If the request to read a NIB attribute was successful, the Status parameter will be set to SUCCESS.
6779 Otherwise, the Status parameter indicates the error.

6780 **3.2.2.28 NLME-SET.request**

6781 This primitive allows the next higher layer to write the value of an attribute into the NIB.

6782 **3.2.2.28.1 Semantics of the Service Primitive**

6783 The semantics of this primitive are as follows:

```
6784 NLME-SET.request      {  
6785   NIBAttribute,  
6786   NIBAttributeLength,  
6787   NIBAttributeValue  
6788 }
```

6789

6790 Table 3.31 specifies the parameters for this primitive.

6791 **Table 3.31 NLME-SET.request Parameters**

Name	Type	Valid Range	Description
NIBAttribute	Integer	See Table 3.49.	The identifier of the NIB attribute to be written.
NIBAttributeLength	Integer	0x0000 – 0xffff	The length, in octets, of the attribute value being set.
NIBAttributeValue	Various	Attribute specific (see Table 3.49)	The value of the NIB attribute that should be written.

6792 **3.2.2.28.2 When Generated**

6793 This primitive is to be generated by the next higher layer and issued to its NLME in order to write the value
6794 of an attribute in the NIB.

6795 **3.2.2.28.3 Effect on Receipt**

6796 On receipt of this primitive the NLME attempts to write the given value to the indicated NIB attribute in its
6797 database. If the NIBAttribute parameter specifies an attribute that is not found in the database, the NLME
6798 issues the NLME-SET.confirm primitive with a status of UNSUPPORTED_ATTRIBUTE. If the NIBAt-
6799 tributeValue parameter specifies a value that is out of the valid range for the given attribute, the NLME is-
6800 sues the NLME-SET.confirm primitive with a status of INVALID_PARAMETER.

6801 If the requested NIB attribute is successfully written, the NLME issues the NLME-SET.confirm primitive
6802 with a status of SUCCESS.

6803 **3.2.2.29 NLME-SET.confirm**

6804 This primitive reports the results of an attempt to write a value to a NIB attribute.

6805 **3.2.2.29.1 Semantics of the Service Primitive**

6806 The semantics of this primitive are as follows:

6807 NLME-SET.confirm {
6808 Status,
6809 NIBAttribute
6810 }

6812 Table 3.32 specifies the parameters for this primitive.

6813 **Table 3.32 NLME-SET.confirm Parameters**

Name	Type	Valid Range	Description
Status	Enumeration	SUCCESS, INVALID_PARAMETER or UNSUPPORTED_ATTRIBUTE	The result of the request to write the NIB attribute.
NIBAttribute	Integer	See Table 3.49.	The identifier of the NIB attrib- ute that was written.

6814

3.2.2.29.2 When Generated

6816 This primitive is generated by the NLME and issued to its next higher layer in response to an
6817 NLME-SET.request primitive. This primitive returns a status of either SUCCESS, indicating that the re-
6818 quired value was written to the indicated NIB attribute, or an error code of INVALID_PARAMETER or
6819 UNSUPPORTED_ATTRIBUTE. The reasons for these status values are fully described in section
6820 3.2.2.28.3.

3.2.2.29.3 Effect on Receipt

6822 On receipt of this primitive, the next higher layer is notified of the results of its request to write the value of
6823 a NIB attribute. If the requested value was written to the indicated NIB attribute, the Status parameter will
6824 be set to SUCCESS. Otherwise, the Status parameter indicates the error.

3.2.2.30 NLME-NWK-STATUS.indication

6826 This primitive allows the next higher layer to be notified of network failures.

3.2.2.30.1 Semantics of the Service Primitive

6828 The semantics of this primitive are as follows:

```
6829 NLME-NWK-STATUS.indication {
6830     Status,
6831     NetworkAddr
6832 }
```

6833
6834 Table 3.33 specifies the parameters for this primitive.

6835 **Table 3.33 NLME-NWK-STATUS.indication Parameters**

Name	Type	Valid Range	Description
Status	Status	Any network status code (see Table 3.42)	The error code associated with the failure.
NetworkAddr	Integer	0x0000 – 0xffff	The 16-bit network address of the device associated with the status information.

6836 **3.2.2.30.2 When Generated**

6837 This primitive is generated by the NWK layer on a device and passed to the next higher layer when one of
6838 the following occurs:

- 6839
 - 6840 • The device has failed to discover or repair a route to the destination given by the NetworkAddr parameter.
 - 6841 • The NWK layer on that device has failed to deliver a frame to its end device child with the 16-bit net-
6842 work address given by the NetworkAddr parameter, due to one of the reasons given in Table 3.42.
 - 6843 • The NWK layer has received a network status command frame destined for the device. In this case, the
6844 values of the NetworkAddr and Status parameters will reflect the values of the destination address and
6845 error code fields of the command frame.

6846 **3.2.2.30.3 Effect on Receipt**

6847 The next higher layer is notified of a failure to communicate with the identified device.

6848 **3.2.2.31 NLME-ROUTE-DISCOVERY.request**

6849 This primitive allows the next higher layer to initiate route discovery.

6850 **3.2.2.31.1 Semantics of the Service Primitive**

6851 The semantics of this primitive are as follows:

```
6852       NLME-ROUTE-DISCOVERY.request     {  
6853               DstAddrMode,  
6854               DstAddr,  
6855               Radius  
6856               NoRouteCache  
6857       }
```

6858
6859 Table 3.34 specifies the parameters for this primitive.

6860 **Table 3.34 NLME-ROUTE-DISCOVERY.request Parameters**

Name	Type	Valid Range	Description
DstAddrMode	Integer	0x00 – 0x02	A parameter specifying the kind of destination address provided. The DstAddrMode parameter may take one of the following three values: 0x00 = No destination address 0x01 = 16-bit network address of a multicast group 0x02 = 16-bit network address of an individual device

Name	Type	Valid Range	Description
DstAddr	16-bit network address	Any network address or multicast address	<p>The destination of the route discovery.</p> <p>If the DstAddrMode parameter has a value of 0x00 then no DstAddr will be supplied. This indicates that the route discovery will be a many-to-one discovery with the device issuing the discovery command as a target.</p> <p>If the DstAddrMode parameter has a value of 0x01, indicating multicast route discovery then the destination will be the 16-bit network address of a multicast group.</p> <p>If the DstAddrMode parameter has a value of 0x02, this indicates unicast route discovery. The DstAddr will be the 16-bit network address of a device to be discovered.</p>
Radius	Integer	0x00 – 0xff	This optional parameter describes the number of hops that the route request will travel through the network.
NoRouteCache	Boolean	TRUE or FALSE	<p>In the case where DstAddrMode has a value of zero, indicating many-to-one route discovery, this flag determines whether the NWK should establish a route record table.</p> <p>TRUE = no route record table should be established FALSE = establish a route record table</p>

6861

3.2.2.31.2 When Generated

6862
6863

This primitive is generated by the next higher layer of a ZigBee coordinator or router and issued to its NLME to request the initiation of route discovery.

6864

3.2.2.31.3 Effect on Receipt

6865
6866
6867

On receipt of this primitive by the NLME of a ZigBee end device, the NLME will issue the NLME-ROUTE-DISCOVERY.confirm primitive to the next higher layer with a status value of INVAILID_REQUEST.

6868
6869
6870
6871

On receipt of this primitive by the NLME with the DstAddrMode parameter not equal to 0x00 and the DstAddr parameter equal to a broadcast address, the NLME will issue the NLME-ROUTE-DISCOVERY.confirm primitive to the next higher layer with a status value of INVAILID_REQUEST.

6872
6873
6874
6875

On receipt of this primitive by the NLME of a ZigBee router or ZigBee coordinator with no routing capacity and with the DstAddrMode parameter equal to 0x01 or 0x02, the NLME will issue the NLME-ROUTE-DISCOVERY.confirm to the next higher layer with a status value of ROUTE_ERROR and a NetworkStatusCode value of 0x04 indicating no routing capacity.

6876
6877
6878
6879

On receipt of this primitive by a ZigBee router or ZigBee coordinator that has routing capacity, with the DstAddrMode parameter equal to 0x02, the NLME will initiate discovery of a unicast route between the current device and the network device with the 16-bit network address given by the DstAddr parameter. The procedure for initiating discovery of a unicast route is outlined in section 3.6.3.5.1.

6880 On receipt of this primitive by a ZigBee router or ZigBee coordinator that has routing capacity, with the
6881 DstAddrMode parameter equal to 0x01, the NLME will first check to see if the device is a member of the
6882 multicast group identified by the DstAddr parameter by checking if the *nwkGroupIDTable* attribute of the
6883 NIB contains an entry corresponding to the destination address. If the device is a member of the multicast
6884 group, then the NLME will immediately issue the NLME-ROUTE-DISCOVERY.confirm primitive with a
6885 status value of SUCCESS and discontinue further processing of the NLME-ROUTE-DISCOVERY.request
6886 primitive. If the device is not a member of the multicast group, the NLME will initiate discovery of a
6887 unicast route between the current device and the multicast group identified by the DstAddr parameter.

6888 On receipt of this primitive on a ZigBee router or ZigBee coordinator with the DstAddrMode parameter
6889 equal to 0x00, the NLME will initiate many-to-one route discovery. The procedure for initiating
6890 many-to-one route discovery is outlined in section 3.6.3.5.1.

6891 In each of the three cases of actual route discovery described above, the NLME will initiate route discovery
6892 by attempting to transmit a route discovery command frame using the MCPS-DATA.request primitive of
6893 the MAC sub-layer. If a value has been supplied for the optional Radius parameter, that value will be
6894 placed in the Radius field of the NWK header of the outgoing frame. If a value has not been supplied then
6895 the radius field of the NWK header will be set to twice the value of the *nwkMaxDepth* attribute of the NIB
6896 as would be the case for data frame transmissions. If the MAC sub-layer fails, for any reason, to transmit
6897 the route request command frame, the NLME will issue the ROUTE-DISCOVERY.confirm primitive to
6898 the next higher layer with a Status parameter value equal to that returned by the MCPS-DATA.confirm. If
6899 the route discovery command frame is sent successfully and if the DstAddrMode parameter has a value of
6900 0x00, indicating many-to-one route discovery, the NLME will immediately issue the
6901 ROUTE-DISCOVERY.confirm primitive with a value of SUCCESS. Otherwise, the NLME will wait until
6902 either a route reply command frame is received or the route discovery operation times out as described in
6903 section 3.6.3.5. If a route reply command frame is received before the route discovery operation times out,
6904 the NLME will issue the NLME-ROUTE-DISCOVERY.confirm primitive to the next higher layer with a
6905 status value of SUCCESS. If the operation times out, it will issue the
6906 NLME_ROUTE-DISCOVERY.confirm primitive with a Status of ROUTE_ERROR and with a Network-
6907 StatusCode value reflecting the reason for failure as described in Table 3.42.

6908 **3.2.2.32 NLME_ROUTE-DISCOVERY.confirm**

6909 This primitive informs the next higher layer about the results of an attempt to initiate route discovery.

6910 **3.2.2.32.1 Semantics of the Service Primitive**

6911 The semantics of this primitive are as follows:

6912 NLME_ROUTE-DISCOVERY.confirm {
6913 Status,
6914 NetworkStatusCode
6915 }

6916
6917 Table 3.35 specifies the parameters for the NLME-ROUTE-DISCOVERY.confirm primitive.

Table 3.35 NLME_ROUTE-DISCOVERY.confirm Parameters

Name	Type	Valid Range	Description
Status	status value	INVALID_REQUEST, ROUTE_ERROR or any status value returned by the MCPS-DATA.confirm primitive	The status of an attempt to ini- tiate route discovery.
NetworkStatusCode	Network status code	See Table 3.42.	In the case where the Status parameter has a value of ROUTE_ERROR, this code gives further information about the kind of error that occurred. Otherwise, it should be ig- nored.

6919

3.2.2.32.2 When Generated

6920
6921

This primitive is generated by the NLME and passed to the next higher layer as a result of an attempt to in-
itiate route discovery.

6922

3.2.2.32.3 Effect on Receipt

6923
6924
6925

The next higher layer is informed of the status of its attempt to initiate route discovery. Possible values for
the Status parameter and the circumstances under which they are generated are described in section
3.2.2.32.3.

6926

3.3 Frame Formats

6927
6928

This section specifies the format of the NWK frame (NPDU). Each NWK frame consists of the following
basic components:

6929
6930

- A NWK header, which comprises frame control, addressing and sequencing information
- A NWK payload, of variable length, which contains information specific to the frame type

6931
6932
6933
6934
6935
6936

The frames in the NWK layer are described as a sequence of fields in a specific order. All frame formats in
this section are depicted in the order in which they are transmitted by the MAC sub-layer, from left to right,
where the leftmost bit is transmitted first. Bits within each field are numbered from 0 (leftmost and least
significant) to k-1 (rightmost and most significant), where the length of the field is k bits. Fields that are
longer than a single octet are sent to the MAC sub-layer in the order from the octet containing the low-
est-numbered bits to the octet containing the highest-numbered bits.

6937

3.3.1 General NPDU Frame Format

6938
6939

The NWK frame format is composed of a NWK header and a NWK payload. The fields of the NWK head-
er appear in a fixed order. The NWK frame shall be formatted as illustrated in Figure 3.5.

6940

Figure 3.5 General NWK Frame Format

Octets: 2	2	2	1	1	0/8	0/8	0/1	Variable	Variable
Frame control	Destination address	Source address	Radius	Sequence number	Destination IEEE Address	Source IEEE Address	Multicast control	Source route subframe	Frame payload
NWK Header									Payload

6941

3.3.1.1 Frame Control Field

6942
6943
6944

The frame control field is 16 bits in length and contains information defining the frame type, addressing and sequencing fields and other control flags. The frame control field shall be formatted as illustrated in Figure 3.6.

6945

Figure 3.6 Frame Control Field

Bits: 0-1	2-5	6-7	8	9	10	11	12	13	14-15
Frame type	Protocol version	Discover route	Multicast flag	Security	Source Route	Destination IEEE Address	Source IEEE Address	End Device Initiator	Reserved

6946
6947
6948

Table 3.36 shows the allowable frame control sub-field configurations for NWK data frames. Note that all frames listed below will have a frame type sub-field equal to 00 indicating data and a protocol version sub-field reflecting the version of the ZigBee specification implemented.

6949

Table 3.36 Allowable Frame Control Sub-Field Configurations

Data Transmission Method	Discover Route	Multicast	Security	Destination IEEE Address	Source IEEE Address
Unicast	00 or 01	0	0 or 1	0 or 1	0 or 1
Broadcast	00	0	0 or 1	0	0 or 1
Multicast	00	1	0 or 1	0	0 or 1
Source routed	00	0	0 or 1	0 or 1	0 or 1

6950

3.3.1.1.1 Frame Type Sub-Field

6951
6952

The frame type sub-field is 2 bits in length and shall be set to one of the non-reserved values listed in Table 3.37.

6953

Table 3.37 Values of the Frame Type Sub-Field

Frame Type Value $b_1\ b_0$	Frame Type Name
00	Data
01	NWK command
10	Reserved
11	Inter-PAN

6954

3.3.1.1.2 Protocol Version Sub-Field

6955 The protocol version sub-field is 4 bits in length and shall be set to a number reflecting the ZigBee NWK
6956 protocol version in use. The protocol version in use on a particular device shall be made available as the
6957 value of the NWK constant *nwkProtocolVersion*.

6958

3.3.1.1.3 Discover Route Sub-Field

6959 The discover route sub-field may be used to control route discovery operations for the transit of this frame
6960 (see section 3.6.3.5).

6961

Table 3.38 Values of the Discover Route Sub-Field

Discover Route Field Value	Field Meaning
0x00	Suppress route discovery
0x01	Enable route discovery
0x02, 0x03	Reserved

6962

6963 For NWK layer command frames, the discover route sub-field shall be set to 0x00 indicating suppression of
6964 route discovery.

6965

3.3.1.1.4 Multicast Flag Sub-Field

6966 The multicast flag sub-field is 1 bit in length and has the value 0 if the frame is a unicast or broadcast frame
6967 and the value 1 if it is a multicast frame. The multicast control field of the NWK header shall be present
6968 only if the multicast flag has the value 1.

6969

3.3.1.1.5 Security Sub-Field

6970 The security sub-field shall have a value of 1 if, and only if, the frame is to have NWK security operations
6971 enabled. If security for this frame is implemented at another layer or disabled entirely, it shall have a value
6972 of 0.

6973

3.3.1.1.6 Source Route Sub-Field

6974 The source route sub-field shall have a value of 1 if and only if a source route subframe is present in the
6975 NWK header. If the source route subframe is not present, the source route sub-field shall have a value of 0.

6976 **3.3.1.1.7 Destination IEEE Address Sub-Field**

6977 The destination IEEE address sub-field shall have a value of 1 if, and only if, the NWK header is to include
6978 the full IEEE address of the destination.

6979 **3.3.1.1.8 Source IEEE Address Sub-Field**

6980 The source IEEE address sub-field shall have a value of 1 if, and only if, the NWK header is to include the
6981 full IEEE address of the source device.

6982 **3.3.1.1.9 End Device Initiator**

6983 If the source of the message is an end device and the *nwkParentInformation* field of the NIB is a value other
6984 than 0, then this sub-field shall be set to 1. Otherwise this sub-field shall be set to 0. After validating
6985 the source (see section 3.6.2.2), a router parent device shall clear this field when relaying a message sent by
6986 one of its end device children.

6987 **3.3.1.2 Destination Address Field**

6988 The destination address field shall always be present and shall be 2 octets in length. If the multicast flag
6989 sub-field of the frame control field has the value 0, the destination address field shall hold the 16-bit net-
6990 work address of the destination device or a broadcast address (see Table 3.59). If the multicast flag
6991 sub-field has the value 1, the destination address field shall hold the 16-bit Group ID of the destination
6992 multicast group. Note that the network address of a device shall be set to the value of the *macShortAddress*
6993 attribute of the MAC PIB.

6994 **3.3.1.3 Source Address Field**

6995 The source address field shall always be present. It shall always be 2 octets in length and shall hold the
6996 network address of the source device of the frame. Note that the network address of a device shall be set to
6997 value of the *macShortAddress* attribute of the MAC PIB.

6998 **3.3.1.4 Radius Field**

6999 The radius field shall always be present. It will be 1 octet in length and specifies the range of a radi-
7000 us-limited transmission. The field shall be decremented by 1 by each receiving device.

7001 **3.3.1.5 Sequence Number Field**

7002 The sequence number field is present in every frame and is 1 octet in length. The sequence number value
7003 shall be incremented by 1 with each new frame transmitted. The values of the source address and sequence
7004 number fields of a frame, taken as a pair, may be used to uniquely identify a frame within the constraints
7005 imposed by the sequence number's one-octet range. For more details on the use of the sequence number
7006 field, see section 3.6.2.

7007 **3.3.1.6 Destination IEEE Address Field**

7008 The destination IEEE address field, if present, contains the 64-bit IEEE address corresponding to the 16-bit
7009 network address contained in the destination address field of the NWK header. Upon receipt of a frame
7010 containing a 64-bit IEEE address, the contents of the *nwkAddressMap* and neighbor table should be
7011 checked for consistency, and updated if necessary. Section 3.6.1.9.2 describes the actions to take in detect-
7012 ing address conflicts. If the 16-bit network address is a broadcast or multicast address then the destination
7013 IEEE address field shall not be present.

7014 **3.3.1.7 Source IEEE Address Field**

7015 The source IEEE address field, if present, contains the 64-bit IEEE address corresponding to the 16-bit
7016 network address contained in the source address field of the NWK header. Upon receipt of a frame con-
7017 taining a 64-bit IEEE address, the contents of the *nwkAddressMap* and Neighbor Table should be checked
7018 for consistency, and updated if necessary. Section 3.6.1.9.2 describes the actions to take in detecting ad-
7019 dress conflicts.

7020 **3.3.1.8 Multicast Control Field**

7021 The multicast control sub-field is 1 octet in length and shall only be present if the multicast flag sub-field
7022 has a value of 1. It is divided into three sub-fields as illustrated in Figure 3.7.

7023 **Figure 3.7 Multicast Control Field Format**

Bits: 0 – 1	2 – 4	5 – 7
Multicast mode	NonmemberRadius	MaxNonmemberRadius

7024 **3.3.1.8.1 Multicast Mode Sub-Field**

7025 The multicast mode sub-field indicates whether the frame is to be transmitted using member or
7026 non-member mode. Member mode is used to propagate multicasts between the devices that are members of
7027 the destination group. Non-member mode is used to transmit a multicast frame from a device that is not a
7028 member of the multicast group to a device that is a member of the multicast group.

7029 **Table 3.39 Values of the Multicast Mode Sub-Field**

Multicast Mode Field Value	Field Meaning
0x0	Non-member mode
0x1	Member mode
0x2	Reserved
0x3	Reserved

7030 **3.3.1.8.2 NonmemberRadius Sub-Field**

7031 The nonmemberradius sub-field indicates the range of a member mode multicast when relayed by devices
7032 that are not members of the destination group. Receiving devices that are members of the destination group
7033 will set this field to the value of the MaxNonmemberRadius sub-field. The originating device and receiving
7034 devices that are not members of the destination group will discard the frame if the NonmemberRadius
7035 sub-field has value 0 and will decrement the field if the NonmemberRadius sub-field has a value in the
7036 range 0x01 through 0x06. A value of 0x07 indicates an infinite range and is not decremented.

7037 The value of the NonmemberRadius sub-field will never exceed the value of the MaxNonmemberRadius
7038 sub-field.

7039 **3.3.1.8.3 MaxNonmemberRadius Sub-Field**

7040 The maximum value of the NonmemberRadius sub-field for this frame.

7041 **3.3.1.9 Source Route Subframe Field**

7042 The source route subframe field shall only be present if the source route sub-field of the frame control field
7043 has a value of 1. It is divided into three sub-fields as illustrated in Figure 3.8.

7044 **Figure 3.8 Source Route Subframe Format**

Octets: 1	1	Variable
Relay count	Relay index	Relay list

7045 **3.3.1.9.1 Relay Count Sub-Field**

7046 The relay count sub-field indicates the number of relays contained in the relay list sub-field of the source
7047 route subframe.

7048 **3.3.1.9.2 Relay Index**

7049 The relay index sub-field indicates the index of the next relay in the relay list sub-field to which the packet
7050 will be transmitted. This field is initialized to 1 less than the relay count by the originator of the packet, and
7051 is decremented by 1 by each receiving relay.

7052 **3.3.1.9.3 Relay List Sub-Field**

7053 The relay list sub-field shall contain the list of relay addresses. The relay closest to the destination shall be
7054 listed first. The relay closest to the originator shall be listed last.

7055 **3.3.1.9.4 Frame Payload Field**

7056 The frame payload field has a variable length and contains information specific to individual frame types.

7057 **3.3.2 Format of Individual Frame Types**

7058 There are two defined NWK frame types: data and NWK command. Each of these frame types is discussed
7059 in the following sections.

7060 **3.3.2.1 Data Frame Format**

7061 The data frame shall be formatted as illustrated in Figure 3.9.

7062 **Figure 3.9 Data Frame Format**

Octets: 2	Variable	Variable
Frame control	Routing fields	Data payload
NWK header		NWK payload

7063

7064 The order of the fields of the data frame shall conform to the order of the general NWK frame format as il-
7065 lustrated in Figure 3.5.

7066 **3.3.2.1.1 Data Frame NWK Header Field**
7067 The data frame NWK header field shall contain the frame control field and an appropriate combination of
7068 routing fields as required.

7069 In the frame control field, the frame type sub-field shall contain the value that indicates a data frame, as
7070 shown in Table 3.37. All other sub-fields shall be set according to the intended use of the data frame.

7071 The routing fields shall contain an appropriate combination of address and broadcast fields, depending on
7072 the settings in the frame control field (see Figure 3.6).

7073 **3.3.2.1.2 Data Payload Field**

7074 The data frame data payload field shall contain the sequence of octets that the next higher layer has re-
7075 quested the NWK layer to transmit.

7076 **3.3.2.2 NWK Command Frame Format**

7077 The NWK command frame shall be formatted as illustrated in Figure 3.10.

7078 **Figure 3.10 NWK Command Frame Format**

Octets: 2	Variable	1	Variable
Frame control	Routing fields	NWK command identifier	NWK command payload
NWK header		NWK payload	

7079
7080 The order of the fields of the NWK command frame shall conform to the order of the general NWK frame
7081 as illustrated in Figure 3.5.

7082 **3.3.2.2.1 NWK Command Frame NWK Header Field**

7083 The NWK header field of a NWK command frame shall contain the frame control field and an appropriate
7084 combination of routing fields as required.

7085 In the frame control field, the frame type sub-field shall contain the value that indicates a NWK command
7086 frame, as shown in Table 3.37. All other sub-fields shall be set according to the intended use of the NWK
7087 command frame.

7088 The routing fields shall contain an appropriate combination of address and broadcast fields, depending on
7089 the settings in the frame control field.

7090 **3.3.2.2.2 NWK Command Identifier Field**

7091 The NWK command identifier field indicates the NWK command being used. This field shall be set to one
7092 of the non-reserved values listed in Table 3.40.

7093 **3.3.2.2.3 NWK Command Payload Field**

7094 The NWK command payload field of a NWK command frame shall contain the NWK command itself.

7095 **3.4 Command Frames**

7096 The command frames defined by the NWK layer are listed in Table 3.40. The following sections detail how
7097 the NLME shall construct the individual commands for transmission.

7098 For each of these commands, the following applies to the NWK header fields unless specifically noted in
7099 the clause on NWK header in each command:

- 7100 • The frame type sub-field of the NWK frame control field shall be set to indicate that this frame is a
7101 NWK command frame.
7102 • The discover route sub-field in the NWK header shall be set to suppress route discovery (see Table
7103 3.38).
7104 • The source address field in the NWK header shall be set to the address of the originating device.

7105 **Table 3.40 NWK Command Frames**

Command Frame Identifier	Command Name	Reference
0x01	Route request	3.4.1
0x02	Route reply	3.4.2
0x03	Network Status	3.4.3
0x04	Leave	3.4.4
0x05	Route Record	3.4.5
0x06	Rejoin request	3.4.6
0x07	Rejoin response	3.4.7
0x08	Link Status	3.4.8
0x09	Network Report	3.4.9
0x0a	Network Update	3.4.10
0x0b	End Device Timeout Request	3.4.11
0x0c	End Device Timeout Re- sponse	3.4.12
0x0d – 0xff	Reserved	—

7106 **3.4.1 Route Request Command**

7107 The route request command allows a device to request other devices within radio range to engage in a
7108 search for a particular destination device and establish a state within the network that will allow messages
7109 to be routed to that destination more easily and economically in the future. The payload of a route request
7110 command shall be formatted as illustrated in Figure 3.11.

7111

Figure 3.11 Route Request Command Frame Format

Octets: 1	1	2	1	0/8
Command options	Route request identifier	Destination address	Path cost	Destination IEEE Address
NWK command payload				

7112

3.4.1.1 MAC Data Service Requirements

7113
7114

In order to transmit this command using the MAC data service, specified in IEEE 802.15.4-2003 [B1], the following information shall be included in the MAC frame header:

7115
7116

- The destination PAN identifier shall be set to the PAN identifier of the device sending the route request command.
- The destination address shall be set to the broadcast address of 0xffff.
- The source address and PAN identifier shall be set to the network address and PAN identifier of the device sending the route request command, which may or may not be the device from which the command originated.
- The frame control field shall be set to specify that the frame is a MAC data frame with MAC security disabled, since any secured frame originating from the NWK layer shall use NWK layer security. Because the frame is broadcast, no acknowledgment request shall be specified.
- The addressing mode and intra-PAN flags shall be set to support the addressing fields described here.

7117

7118

7119
7120

7121
7122
7123

7124

7125

3.4.1.2 NWK Header Fields

7126
7127

In order for this route request to reach its destination and for the route discovery process to complete correctly, the following information must be provided:

7128
7129

7130
7131
7132

- The destination address in the NWK header shall be set to the broadcast address for all routers and the coordinator (see Table 3.59).
- The source IEEE address sub-field of the frame control field shall be set to 1 and the source IEEE address field of the NWK header shall be present and shall contain the 64-bit IEEE address of the originator of the frame.

7133

3.4.1.3 NWK Payload Fields

7134
7135
7136

The NWK frame payload contains a command identifier field, a command options field, the route request identifier field, the address of the intended destination, an up-to-date summation of the path cost, and the destination IEEE address.

7137

The command frame identifier shall contain the value indicating a route request command frame.

7138

3.4.1.3.1 Command Options Field

7139

The format of the 8-bit command options field is shown in Figure 3.12.

7140

Figure 3.12 Route Request Command Options Field

Bit: 0-2	3-4	5	6	7
Reserved	Many-to-one	Destination IEEE address	Multicast	Reserved

7141 **3.4.1.3.1.1 Many-to-One**

7142 The many-to-one field shall have one of the non-reserved values shown in Table 3.41.

7143 **Table 3.41 Many-to-One Field Values**

Value	Description
0	The route request is not a many-to-one route request.
1	The route request is a many-to-one route request and the sender supports a route record table.
2	The route request is a many-to-one route request and the sender does not support a route record table.
3	Reserved

7144 **3.4.1.3.1.2 Destination IEEE Address**

7145 The destination IEEE address field is a single-bit field. It shall have a value of 1 if, and only if, the command frame contains the destination IEEE address. The Destination IEEE Address field should always be added if it is known.

7148 **3.4.1.3.1.3 Multicast Sub-Field**

7149 The multicast sub-field is a single-bit field. It shall have a value of 1 if, and only if, the command frame is a request for a route to a multicast group, in which case the destination address field contains the Group ID of the desired group.

7152 **3.4.1.3.2 Route Request Identifier**

7153 The route request identifier is an 8-bit sequence number for route requests and is incremented by 1 every time the NWK layer on a particular device issues a route request.

7155 **3.4.1.3.3 Destination Address**

7156 The destination address shall be 2 octets in length and represents the intended destination of the route request command frame.

7158 **3.4.1.3.4 Path Cost**

7159 The path cost field is eight bits in length and is used to accumulate routing cost information as a route request command frame moves through the network (see section 3.6.3.5.2).

7161 **3.4.1.3.5 Destination IEEE Address**

7162 The destination IEEE address shall be 8 octets in length and represents the IEEE address of the destination of the route request command frame. It shall be present only if the destination IEEE address sub-field of the command frame options field has a value of 1.

7165

3.4.2 Route Reply Command

7166
7167
7168
7169
7170

The route reply command allows the specified destination device of a route request command to inform the originator of the route request that the request has been received. It also allows ZigBee routers along the path taken by the route request to establish state information that will enable frames sent from the source device to the destination device to travel more efficiently. The payload of the route reply command shall be formatted as illustrated in Figure 3.13.

7171

Figure 3.13 Route Reply Command Format

Octets: 1	1	2	2	1	0/8	0/8
Command options	Route request identifier	Originator address	Responder address	Path cost	Originator IEEE address	Responder IEEE address
NWK command payload						

7172

3.4.2.1 MAC Data Service Requirements

7173
7174

In order to transmit this command using the MAC data service, specified in IEEE 802.15.4-2003 [B1], the following information shall be included in the MAC frame header:

7175
7176
7177
7178

- The destination MAC address and PAN identifier shall be set to the network address and PAN identifier, respectively, of the first hop in the path back to the originator of the corresponding route request command frame. The destination PAN identifier shall be the same as the PAN identifier of the originator.
- The source MAC address and PAN identifier shall be set to the network address and PAN identifier of the device sending the route reply command, which may or may not be the device from which the command originated.
- The frame control field shall be set to specify that the frame is a MAC data frame with MAC security disabled, since any secured frame originating from the NWK layer shall use NWK layer security. The transmission options shall be set to require acknowledgment. The addressing mode and intra-PAN flags shall be set to support the addressing fields described here.

7186

3.4.2.2 NWK Header Fields

7187
7188

In order for this route reply to reach its destination and for the route discovery process to complete correctly, the following information must be provided:

7189
7190
7191
7192
7193
7194
7195
7196
7197
7198

- The source address in the NWK header shall be set to the 16-bit network address of the device transmitting the frame.
- The destination address field in the NWK header shall be set to the network address of the first hop in the path back to the originator of the corresponding route request.
- Since this is a NWK layer command frame, the source IEEE address sub-field of the frame control field shall be set to 1 and the source IEEE address field of the NWK header shall be present and shall contain the 64-bit IEEE address of the originator of the frame. The destination IEEE address sub-field of the frame control field shall also have a value of 1 and the destination IEEE address field of the NWK header shall be present and shall contain the 64-bit IEEE address of the first hop in the path back to the originator of the corresponding route request.
- The Sequence Number field in the NWK header shall be created for every hop during the route reply process. The Radius Field shall be set to $nwkMaxDepth * 2$ by the target of the route request. Every hop during the Route Reply process shall decrement the radius by 1. If the value of the radius in the received Route Reply message is 1, the relaying router shall set the radius of the message to 1. The Sequence Number shall be created as if it were a new frame from the device transmitting the frame replacing the sequence number with the device's next available sequence number. The Route Reply frame is not a forwarded frame, but is newly created by each hop during the route reply process.

7206

3.4.2.3 NWK Payload Fields

7207
7208

The NWK frame payload contains a command identifier field, a command options field, the route request identifier, originator and responder addresses and an up-to-date summation of the path cost.

7209 The command frame identifier shall contain the value indicating a route reply command frame.

3.4.2.3.1 Command Options Field

7211 The format of the 8-bit command options field is shown in Figure 3.14.

7212 **Figure 3.14 Route Reply Command Options Field**

Bit: 0 – 3	4	5	6	7
Reserved	Originator IEEE address	Responder IEEE address	Multicast	Reserved

3.4.2.3.1.1 Originator IEEE Address

7214 The originator IEEE address sub-field is a single-bit field. It shall have a value of 1 if and only if the originator IEEE address field is included in the payload. This bit shall be set when *nwkUniqueAddr* is FALSE.

3.4.2.3.1.2 Responder IEEE Address

7217 The responder IEEE address sub-field is a single-bit field. It shall have a value of 1 if, and only if, the responder IEEE address field is included in the payload. This bit shall be set when *nwkUniqueAddr* is FALSE and the multicast sub-field is set to 0.

3.4.2.3.1.3 Multicast Sub-Field

7221 The multicast sub-field is a single-bit field. It shall have a value of 1 if and only if the command frame is a reply to a request for a route to a multicast group, in which case the responder address field contains the Group ID of the desired group.

3.4.2.3.2 Route Request Identifier

7225 The route request identifier is the 8-bit sequence number of the route request to which this frame is a reply.

3.4.2.3.3 Originator Address

7227 The originator address field shall be 2 octets in length and shall contain the 16-bit network address of the originator of the route request command frame to which this frame is a reply.

3.4.2.3.4 Responder Address

7230 The responder address field shall be 2 octets in length and shall always be the same as the value in the destination address field of the corresponding route request command frame.

3.4.2.3.5 Path Cost

7233 The path cost field is used to sum link cost as the route reply command frame transits the network (see section 3.6.3.5.3).

3.4.2.3.6 Originator IEEE Address

7236 The originator IEEE address field shall be 8 octets in length and shall contain the 64-bit address of the originator of the route request command frame to which this frame is a reply. This field shall only be present if the originator IEEE address sub-field of the command options field has a value of 1.

3.4.2.3.7 Responder IEEE Address

7240 The responder IEEE address field shall be 8 octets in length and shall contain the 64-bit address of the destination of the route request command frame to which this frame is a reply. This field shall only be present if the responder IEEE address sub-field of the command options field has a value of 1.

7243

3.4.3 Network Status Command

7244
7245
7246
7247

A device uses the network status command to report errors and other conditions arising in the NWK layer of a particular device to the peer NWK layer entities of other devices in the network. The NWK status command may be also used to diagnose network problems, for example address conflicts. The payload of a network status command shall be formatted as illustrated in Figure 3.15.

7248

Figure 3.15 Network Status Command Frame Format

Octets: 1	2
Status code	Destination address
NWK command payload	

7249

3.4.3.1 MAC Data Service Requirements

7250
7251

In order to transmit this command using the MAC data service, specified in IEEE 802.15.4-2003 [B1], the following information shall be provided:

7252
7253
7254
7255
7256
7257
7258
7259
7260
7261

- The destination MAC address and PAN identifier shall be set to the network address and PAN identifier, respectively, of the first hop in the path to the destination of the command frame or to the broadcast address 0xffff in the case where the command frame is being broadcast at the NWK layer.
- The source MAC address and PAN identifier shall be set to the network address and PAN identifier of the device sending the network status command.
- The frame control field shall be set to specify that the frame is a MAC data frame with MAC security disabled, since any secured frame originating from the NWK layer shall use NWK layer security. The transmission options shall not be set to require acknowledgement if the destination MAC address is the broadcast address 0xffff.
- The addressing mode and intra-PAN flags shall be set to support the addressing fields described here.

7262

3.4.3.2 NWK Header Fields

7263
7264

Network status commands may be either unicast or broadcast. The fields of the NWK header shall be set as follows:

7265
7266
7267
7268
7269
7270
7271
7272
7273
7274
7275

- The source address field shall always be set to the 16-bit network address of the device originating the command frame.
- The source IEEE address sub-field of the frame control field shall be set to 1 and the source IEEE address field of the NWK header shall be present and shall contain the 64-bit IEEE address of the originator of the frame.
- When sent in response to a routing error, the destination address field in the NWK header shall be set to the same value as the source address field of the data frame that encountered a forwarding failure.
- If and only if, the network status command frame is not broadcast, the destination IEEE address sub-field of the frame control field shall have a value of 1 and the destination IEEE address field of the NWK header shall be present and shall contain the 64-bit IEEE corresponding to the 16-bit network address in the destination address field if this IEEE address is known.

7276

3.4.3.3 NWK Payload Fields

7277
7278
7279

The NWK frame payload of the network status command frame contains a command frame identifier field, a status code field and a destination address field as described below. The command frame identifier shall be set to specify the network status command frame as defined in Table 3.40.

7280 **3.4.3.3.1 Status Code**

7281 The status code shall be set to one of the non-reserved values shown in Table 3.42.

7282 **Table 3.42 Status Codes for Network Status Command Frame**

Value	Status Code
0x00	No route available
0x01	Tree link failure
0x02	Non-tree link failure
0x03	Low battery level
0x04	No routing capacity
0x05	No indirect capacity
0x06	Indirect transaction expiry
0x07	Target device unavailable
0x08	Target address unallocated
0x09	Parent link failure
0x0a	Validate route
0x0b	Source route failure
0x0c	Many-to-one route failure
0x0d	Address conflict
0x0e	Verify addresses
0x0f	PAN identifier update
0x10	Network address update
0x11	Bad frame counter
0x12	Bad key sequence number
0x13 – 0xff	Reserved

7283

7284 These status codes are used both as values for the status code field of a network status command frame and
7285 as values of the Status parameter of the NLME-NWK-STATUS.indication primitive. A brief explanation of
7286 each follows:

- 7287 • **No route available:** Route discovery and/or repair has been attempted and no route to the intended
7288 destination address has been discovered.
- 7289 • **Tree link failure:** The routing failure occurred as a result of the failure of an attempt to route the frame
7290 along the tree.
- 7291 • **Non-tree link failure:** The routing failure did not occur as a result of an attempt to route along the
7292 tree.
- 7293 • **Low battery level:** The frame was not relayed because the relaying device was running low on battery
7294 power.
- 7295 • **No routing capacity:** The failure occurred because the relaying device has no routing capacity.
- 7296 • **No indirect capacity:** The failure occurred as the result of an attempt to buffer a frame for a sleeping
7297 end device child and the relaying device had no buffer capacity to use.
- 7298 • **Indirect transaction expiry:** A frame that was buffered on behalf of a sleeping end device child has
7299 been dropped as a result of a time-out.
- 7300 • **Target device unavailable:** An end device child of the relaying device is for some reason unavailable.
- 7301 • **Target address unallocated:** The frame was addressed to a non-existent end device child of the re-
7302 laying device.
- 7303 • **Parent link failure:** The failure occurred as a result of a failure in the RF link to the device's parent.
7304 This status is only used locally on a device to indicate loss of communication with the parent.
- 7305 • **Validate route:** The multicast route identified in the destination address field should be validated as
7306 described in section 3.6.3.6.
- 7307 • **Source route failure:** Source routing has failed, probably indicating a link failure in one of the source
7308 route's links.
- 7309 • **Many-to-one route failure:** A route established as a result of a many-to-one route request has failed.
- 7310 • **Address conflict:** The address in the destination address field has been determined to be in use by two
7311 or more devices.
- 7312 • **Verify addresses:** The source device has the IEEE address in the Source IEEE address field and, if the
7313 Destination IEEE address field is present, the value it contains is the expected IEEE address of the des-
7314 tination.
- 7315 • **PAN identifier update:** The operational network PAN identifier of the device has been updated.
- 7316 • **Network address update:** The network address of the device has been updated.
- 7317 • **Bad frame counter:** A frame counter reported in a received frame had a value less than or equal to
7318 that stored in *nwkSecurityMaterialSet*.
- 7319 • **Bad key sequence number:** The key sequence number reported in a received frame did not match that
7320 of *nwkActiveKeySeqNumber*.

7321 3.4.3.3.2 Destination Address

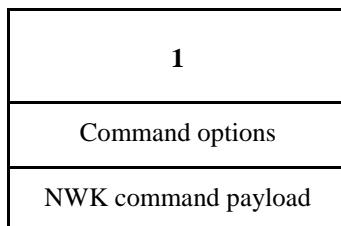
7322 The destination address field is 2 octets in length and shall be present if, and only if, the network status
7323 command frame is being sent in response to a routing failure. In this case, it shall contain the destination
7324 address from the data frame that encountered the failure.

7325 3.4.4 Leave Command

7326 The leave command is used by the NLME to inform other devices on the network that a device is leaving
7327 the network or else to request that a device leave the network. The payload of the leave command shall be
7328 formatted as shown in Figure 3.16.

7329

Figure 3.16 Leave Command Frame Format



7330 **3.4.4.1 MAC Data Service Requirement**

7331 In order to transmit this command using the MAC data service, specified in IEEE 802.15.4-2003 [B1], the
7332 following information shall be provided:

- 7333 • The destination MAC address and PAN identifier shall be set to the network address and PAN identi-
7334 fier, respectively, of the neighbor device to which the frame is being sent or else to the MAC broadcast
7335 address 0xffff in the case where the NWK header also contains a broadcast address.
- 7336 • The source MAC address and PAN identifier shall be set to the network address and PAN identifier of
7337 the device sending the leave command.
- 7338 • The frame control field shall be set to specify that the frame is a MAC data frame with MAC security
7339 disabled, since any secured frame originating from the NWK layer shall use NWK layer security. Ac-
7340 knowledgment shall be requested.
- 7341 • The addressing mode and intra-PAN flags shall be set to support the addressing fields described here.

7342 **3.4.4.2 NWK Header Fields**

7343 The NWK header fields of the leave command frame shall be set as follows:

- 7344 • The source IEEE address sub-field of the frame control field shall be set to 1 and the source IEEE ad-
7345 dress field of the NWK header shall be present and shall contain the 64-bit IEEE address of the origi-
7346 nator of the frame.
- 7347 • If the request sub-field of the command options field is set to 1 then the destination address field in the
7348 NWK header shall be set to the network address of the child device being requested to leave.
- 7349 • If the request sub-field is set to 0 then the destination address field in the NWK header shall be set to
7350 0xffff so that the indication is received by devices with *macRxOnWhenIdle* equal to TRUE.
- 7351 • The destination address sub-field of the frame control may be set to 0 or 1. The choice shall be based
7352 on whether the local device has knowledge of the IEEE address for the device being requested to leave.
7353 If the local device knows the IEEE address then the field shall be set to 1 and the destination IEEE ad-
7354 dress field shall be present..
- 7355 • The radius field shall be set to 1.

7356 **3.4.4.3 NWK Payload Fields**

7357 The NWK payload of the leave command frame contains a command frame identifier field and a command
7358 options field. The command frame identifier field shall be set to specify the leave command frame as de-
7359 scribed in Table 3.40.

7360 **3.4.4.3.1 Command Options Field**

7361 The format of the 8-bit Command Options field is shown in Figure 3.17.

7362

Figure 3.17 Leave Command Options Field

Bit: 0 – 4	5	6	7
Reserved	Rejoin	Request	Remove children

7363

3.4.4.3.1.1 Rejoin Sub-Field

7364 The Rejoin sub-field is a single-bit field. If the value of this sub-field is 1, the device that is leaving from its
7365 current parent will rejoin the network. If the value of this sub-field is 0, the device will not rejoin the net-
7366 work.

7367

3.4.4.3.1.2 Request Sub-Field

7368 The request sub-field is a single-bit field. If the value of this sub-field is 1, then the leave command frame
7369 is a request for another device to leave the network. If the value of this sub-field is 0, then the leave com-
7370 mand frame is an indication that the sending device plans to leave the network.

7371

3.4.4.3.1.3 Remove Children Sub-Field

7372 The remove children sub-field is a single-bit field. If this sub-field has a value of 1, then the children of the
7373 device that is leaving the network will also be removed. If this sub-field has a value of 0, then the children
7374 of the device leaving the network will not be removed.

7375

3.4.5 Route Record Command

7376 The route record command allows the route taken by a unicast packet through the network to be recorded in
7377 the command payload and delivered to the destination device. The payload of the route record command
7378 shall be formatted as illustrated in Figure 3.18.

7379

Figure 3.18 Route Record Command Format

Octets: 1	Variable
Relay count	Relay list
NWK command payload	

7380

3.4.5.1 MAC Data Service Requirements

7381 In order to transmit this command using the MAC data service, specified in IEEE 802.15.4-2003 [B1], the
7382 following information shall be provided:

7383

- The destination MAC address and PAN identifier shall be set to the network address and PAN identi-
7384 fier, respectively, of the neighbor device to which the frame is being sent.
- The source MAC address and PAN identifier shall be set to the network address and PAN identifier of
7386 the device sending the route record command.
- The frame control field shall be set to specify that the frame is a MAC data frame with MAC security
7388 disabled, since any secured frame originating from the NWK layer shall use NWK layer security. Ac-
7389 knowledgment shall be requested.
- The addressing mode and intra-PAN flags shall be set to support the addressing fields described here.

3.4.5.2 NWK Header Fields

The NWK header fields of the route record command frame shall be set as follows:

- If the route record is being initiated as the result of a NLDE-DATA.request primitive from the next higher layer, the source address field shall be set to the 16-bit network address of the originator of the frame. If the route record is being initiated as a result of the relaying of a data frame on behalf of one of the device's end device children, the source address field shall contain the 16-bit network address of that end device child.
- The source IEEE address sub-field of the frame control field shall be set to 1 and the source IEEE address field of the NWK header shall be present and shall contain the 64-bit IEEE address corresponding to the 16-bit network address contained in the source address field.
- The destination address field in the NWK header shall be set to the 16-bit network address of the concentrator device that is the destination of the frame.
- The destination IEEE address sub-field of the frame control field shall be set to 1, and the destination IEEE address field shall be set to the IEEE address of the concentrator device that is the destination of the frame, if this address is known.
- The Source Route sub-field of the frame control field shall be set to 0.

3.4.5.3 NWK Payload

The NWK frame payload contains a command identifier field, a relay count field, and a relay list field. The command frame identifier shall contain the value indicating a route record command frame.

3.4.5.3.1 Relay Count Field

This field contains the number of relays in the relay list field of the route record command. If the route record is being initiated as the result of a NLDE-DATA.request primitive from the next higher layer, the relay count field is initialized to 0. If the route record is being initiated as a result of the relaying of a data frame on behalf of one of the device's end device children, the relay count field is initialized to 1. In either case, it is incremented by each receiving relay.

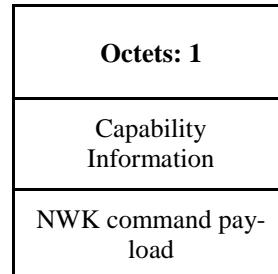
3.4.5.3.2 Relay List Field

The relay list field is a list of the 16-bit network addresses of the nodes that have relayed the packet. If the route record is being initiated as a result of the relaying of a data frame on behalf of one of the device's end device children, the initiating device will initialize this field with its own 16-bit network address. Receiving relay nodes append their network address to the list before forwarding the packet.

3.4.6 Rejoin Request Command

The rejoin request command allows a device to rejoin its network. This is normally done in response to a communication failure, such as when an end device can no longer communicate with its original parent. The rejoin request command shall be formatted as shown in Figure 3.19.

Figure 3.19 Rejoin Request Command Frame Format



3.4.6.1 MAC Data Service Requirements

In order to transmit this command using the MAC data service, specified in IEEE 802.15.4.-2003, [B1], the following information shall be provided:

- The destination address and PAN identifier shall be set to the network address and PAN identifier, respectively, of the prospective parent.
- The source MAC address and PAN identifier shall be set to the network address and PAN identifier of the device transmitting the rejoin command frame.
- The transmission options shall be set to require acknowledgement.
- The addressing mode and intra-PAN flags shall be set to support the addressing fields described here.

3.4.6.2 NWK Header Fields

The NWK header fields of the rejoin request command frame shall be set as follows:

- The source address field of the NWK header to the 16-bit network address shall be as follows. If the value of the *nwkNetworkAddress* in the NIB is within the valid range, then it shall use that value. If the value of the *nwkNetworkAddress* in the NIB is not within the valid range, then it shall randomly generate a value with the valid range, excluding the value of 0x0000, and use that.
- The source IEEE address sub-field of the frame control field shall be set to 1, and the source IEEE address field shall be set to the IEEE address of the device issuing the request.
- The destination address field in the NWK header shall be set to the 16-bit network address of the prospective parent.
- The destination IEEE address sub-field of the frame control field shall be set to 1, and the destination IEEE address field shall be set to the IEEE address of the prospective parent, if this address is known.
- The radius field shall be set to 1.

3.4.6.3 NWK Payload Fields

The NWK frame payload contains a command identifier field and a capability information field. The command frame identifier shall contain the value indicating a rejoin request command frame.

3.4.6.3.1 Capability Information Field

This one-octet field has the format of the capability information field in the association request command in [B1], which is also described in Table 3.52.

3.4.7 Rejoin Response Command

The rejoin response command is sent by a device to inform a child of its network address and rejoin status. The rejoin request command shall be formatted as shown in Figure 3.20.

Figure 3.20 Rejoin Response Command Frame Format

Octets: 2	1
Network address	Rejoin status
NWK command payload	

3.4.7.1 MAC Data Service Requirements

In order to transmit this command using the MAC data service, specified in [B1], the following information shall be provided:

- The destination MAC address and PAN identifier shall be set to the network address and PAN identifier, respectively, of the device that sent the rejoin request to which this frame is a response.
- The source MAC address and PAN identifier shall be set to the network address and PAN identifier of the device that received and processed the rejoin request command frame.
- Acknowledgment shall be requested.
- The addressing mode and intra-PAN flags shall be set to support the addressing fields described here. The TXOptions shall request ‘indirect transmission’ to be used if the *Receiver on when idle* bit of the *nwkCapabilityInformation* contained in the corresponding rejoin request command is equal to 0x00. Otherwise, ‘direct transmission’ shall be used.

3.4.7.2 NWK Header Fields

The NWK header fields of the rejoin response command frame shall be set as follows:

- The source address field shall be set to the 16-bit network address of the device that is sending the response.
- The source IEEE address sub-field of the frame control field shall be set to 1 and the source IEEE address field of the NWK header shall be present and shall contain the 64-bit IEEE address of the parent device that is sending the response.
- The destination address field of the NWK header shall be set to the current network address of the rejoining device, *i.e.* the device that sent the join request to which this frame is a response.
- The destination IEEE address sub-field of the frame control field shall have a value of 1 and the destination IEEE address field of the NWK header shall be present and shall contain the 64-bit IEEE address of the child device that is source of the rejoin request command to which this frame is a response.
- The NWK layer will set the security of the rejoin response command frame to the same level as that of the received rejoin request command frame to which it is a response.

3.4.7.3 NWK Payload Fields

3.4.7.3.1 Network Address Field

If the rejoin was successful, this two-octet field contains the new network address assigned to the rejoining device. If the rejoin was not successful, this field contains the broadcast address (0xffff).

3.4.7.3.2 Rejoin Status Field

This field shall contain one of the non-reserved association status values specified in [B1].

3.4.8 Link Status Command

The link status command frame allows neighboring routers to communicate their incoming link costs to each other as described in section 3.6.3.4. Link status frames are transmitted as one-hop broadcasts without retries.

3.4.8.1 MAC Data Service Requirements

In order to transmit this command using the MAC data service, specified in IEEE 802.15.4-2003 [B1], the following information shall be included in the MAC frame header:

- The destination PAN identifier shall be set to the PAN identifier of the device sending the link status command.

- 7499 • The destination address must be set to the broadcast address of 0xffff.
7500 • The source MAC address and PAN identifier shall be set to the network address and PAN identifier of
7501 the device sending the link status command.
7502 • The frame control field shall be set to specify that the frame is a MAC data frame with MAC security
7503 disabled, since any secured frame originating from the NWK layer shall use NWK layer security. Be-
7504 cause the frame is broadcast, no acknowledgment request shall be specified.
7505 • The addressing mode and intra-PAN flags shall be set to support the addressing fields described here.

3.4.8.2 NWK Header Fields

The NWK header field of the link status command frame shall be set as follows:

- 7508 • The source IEEE address sub-field of the frame control field shall be set to 1 and the source IEEE ad-
7509 dress field of the NWK header shall be present and shall contain the 64-bit IEEE address of the origi-
7510 nator of the frame.
7511 • The destination address in the NWK header shall be set to the router-only broadcast address (see Table
7512 3.59).
7513 • The destination IEEE address sub-field of the frame control field shall have a value of 0 and the desti-
7514 nation IEEE address field of the NWK header shall not be present.
7515 • The radius field shall be set to 1.

3.4.8.3 NWK Payload Fields

The NWK command payload of the link status command shall be formatted as illustrated in Figure 3.21.

Figure 3.21 Link Status Command Format

Octets: 1	Variable
Command options	Link status list
NWK command payload	

3.4.8.3.1 Command Options Field

The format of the 8-bit command options field is shown in Figure 3.22.

Figure 3.22 Link Status Command Options Field

Bit: 0 – 4	5	6	7
Entry count	First frame	Last frame	Reserved

7522 The entry count sub-field of the command options field indicates the number of link status entries present
7523 in the link status list. The first frame sub-field is set to 1 if this is the first frame of the sender's link status.
7524 The last frame sub-field is set to 1 if this is the last frame of the sender's link status. If the sender's link sta-
7525 tus fits into a single frame, the first frame and last frame bits shall both be set to 1.

3.4.8.3.2 Link Status List Field

An entry in the link status list is formatted as shown in Figure 3.23.

7528

Figure 3.23 Link Status Entry

Octets: 2	1
Neighbor network address	Link status

7529 Link status entries are sorted in ascending order by network address. If all router neighbors do not fit in a
7530 single frame, multiple frames are sent. When sending multiple frames, the last network address in the link
7531 status list for frame N is equal to the first network address in the link status list for frame N+1.

7532 Each link status entry contains the network address of a router neighbor, least significant octet first, fol-
7533 lowed by the link status octet. The incoming cost field contains the device's estimate of the link cost for the
7534 neighbor, which is a value between 1 and 7. The outgoing cost field contains the value of the outgoing cost
7535 field from the neighbor table.

7536 The link status field in a link status entry is formatted as follows:

Bits: 0-2	3	4-6	7
Incoming cost	Reserved	Outgoing cost	Reserved

7537 **3.4.9 Network Report Command**

7538 The network report command allows a device to report network events to the device identified by the ad-
7539 dress contained in the *nwkManagerAddr* in the NIB. Such events are radio channel condition and PAN ID
7540 conflicts. The payload of a network report command shall be formatted as illustrated in Figure 3.24.

7541 **Figure 3.24 Network Report Command Frame Format**

Octets: 1	8	Variable
Command op- tions (see Fig- ure 3.25)	EPID	Report in- formation
NWK command payload		

7542 **3.4.9.1 MAC Data Service Requirements**

7543 In order to transmit this command using the MAC data service, specified in [B1], the following information
7544 shall be included in the MAC frame header:

- 7545 • The destination PAN identifier shall be set to the PAN identifier of the device sending the network re-
7546 port command.
- 7547 • The destination address shall be set to the value of the next-hop address field in the routing table entry
7548 for which the destination address field has the same value as the *nwkManagerAddr* field in the NIB. If
7549 no such routing table entry exists, then the NWK may attempt route discovery as described in section
7550 3.6.3.5.

- 7551 • The source MAC address and PAN identifier shall be set to the network address and PAN identifier of
7552 the device sending the network report command, which may or may not be the device from which the
7553 command originated.
7554 • The frame control field shall be set to specify that the frame is a MAC data frame with MAC security
7555 disabled, since any secured frame originating from the NWK layer shall use NWK layer security. The
7556 transmission options shall be set to require acknowledgment.

3.4.9.2 NWK Header Fields

7558 The NWK header fields of the network report command frame shall be set as follows:

- 7559 • The source IEEE address sub-field of the frame control field shall be set to 1 and the source IEEE ad-
7560 dress field of the NWK header shall be present and shall contain the 64-bit IEEE address of the origi-
7561 nator of the frame.
7562 • The destination address field in the NWK header shall be set to the 16-bit network address contained in
7563 the *nwkManagerAddr* attribute of the NIB.
7564 • The destination IEEE address sub-field of the frame control field shall have a value of 1 and the desti-
7565 nation IEEE address field of the NWK header shall be present and shall contain the 64-bit IEEE ad-
7566 dress of the corresponding to the 16-bit network address contained in the *nwkManagerAddr* attribute of
7567 the NIB, if this IEEE address is known.

3.4.9.3 NWK Payload Fields

7569 The NWK frame payload contains a command identifier field, a command options field, an EPID field, and
7570 a report information payload.

7571 The command frame identifier shall contain the value indicating a network report command frame.

3.4.9.3.1 Command Options Field

7573 The format of the 8-bit command options field is shown in Figure 3.25.

7574 **Figure 3.25 Network Report Command Options Field**

Bits 0 - 4	5 - 7
Report information count	Report command identifier (see Figure 3.26)

3.4.9.3.1.1 Report Information Count Sub-Field

7576 The report information count sub-field contains an integer indicating the number of records contained
7577 within the Report Information field. The size of a record depends in the value of the Report Command
7578 Identifier.

3.4.9.3.1.2 Report Command Identifier Sub-Field

7580 The report command identifier sub-field contains an integer indicating the type of report information com-
7581 mand. Figure 3.26 contains the values that can be inserted into this field.

7582

Figure 3.26 Report Command Identifier Sub-Field

Report Command Identifier Value	Report Type
0x00	PAN identifier conflict
0x01 - 0x07	Reserved

7583

3.4.9.3.2 EPID Field

7584
7585

The EPID field shall contain the 64-bit EPID that identifies the network that the reporting device is a member of.

7586

3.4.9.3.3 Report Information

7587
7588

The report information field provides the information being reported, the format of this field depends upon the value of the Report Command Identifier sub-field.

7589

3.4.9.3.3.1 PAN Identifier Conflict Report

7590
7591

If the value of the Report Command Identifier sub-field indicates a PAN identifier conflict report then the Report Information field will have the format shown in Figure 3.27.

7592

Figure 3.27 PAN Identifier Conflict Report

Octets: 2	2	2
1st PAN ID	...	nth PAN ID

7593
7594
7595

The PAN ID conflict report shall be made up of a list of 16-bit PAN identifiers that are operating in the neighborhood of the reporting device. The number of PAN identifiers in the PAN ID conflict report shall be equal to the value of the report information count sub-field of the command options field.

7596

3.4.10 Network Update Command

7597
7598
7599

The network update command allows the device identified by the *nwkManagerAddr* attribute of the NIB to broadcast the change of configuration information to all devices in the network. For example, broadcasting the fact that the network is about to change its short PAN identifier.

7600

The payload of a network update command shall be formatted as illustrated in Figure 3.28.

7601

Figure 3.28 Network Update Command Frame Format

Octets: 1	8	1	Variable
Command Options (see Figure 3.25)	EPID	Update Id	Update Information
NWK command payload			

3.4.10.1 MAC Data Service Requirements

In order to transmit this command using the MAC data service specified in [B1], the following information shall be included in the MAC frame header:

- The destination PAN identifier shall be set to the old PAN identifier of the ZigBee coordinator in order for the command frame to reach network devices which have not received this update. The destination address shall be set according to the procedures for broadcast transmission outlined in section 3.6.5.
- The source MAC address and PAN identifier shall be set to the network address and the old PAN identifier of the device sending the network report command, which may or may not be the device from which the command originated.
- The frame control field shall be set to specify that the frame is a MAC data frame with MAC security disabled, since any secured frame originating from the NWK layer shall use NWK layer security.

3.4.10.2 NWK Header Fields

The NWK header fields of the network update command frame shall be set as follows:

- The source IEEE address sub-field of the frame control field shall be set to 1 and the source IEEE address field of the NWK header shall be present and shall contain the 64-bit IEEE address of the originator of the frame.
- The destination address in the NWK header shall be set to the broadcast address 0xffff.
- The destination IEEE address sub-field of the frame control field shall have a value of 0 and the destination IEEE address field shall not be present in the NWK header.

3.4.10.3 NWK Payload Fields

The NWK frame payload contains a command identifier field, a command options field, an EPID field and an Update Information variable field.

The command frame identifier shall contain the value indicating a network update command frame.

3.4.10.3.1 Command Options Field

The format of the 8-bit command options field is shown in Figure 3.29.

Figure 3.29 Network Update Command Options Field

Bits 0 - 4	5 - 7
Update Information Count	Update Command identifier (see Figure 3.30)

3.4.10.3.1.1 Update Information Count Sub-Field

The update information count sub-field contains an integer indicating the number of records contained within the Update Information field. The size of a record depends on the value of the Update Command Identifier sub-field.

3.4.10.3.1.2 Update Command Identifier Sub-Field

The update command identifier sub-field contains an integer indicating the type of update information command. Figure 3.30 contains the values that can be inserted into this field.

7635

Figure 3.30 Update Command Identifier Sub-Field

Update Command Identifier Value	Report Type
0x00	PAN Identifier Update
0x01 - 0x07	Reserved

7636

3.4.10.3.2 EPID Field

7637

The EPID field shall contain the 64bit EPID that identifies the network that is to be updated.

7638

3.4.10.3.3 Update Id Field

7639
7640

The update Id field will reflect the current value of the *nwkUpdateId* attribute of the device sending the frame.

7641

3.4.10.3.4 Update Information

7642
7643

The update information field provides the information being updated, the format of this field depends upon the value of the Update Command Identifier sub-field.

7644

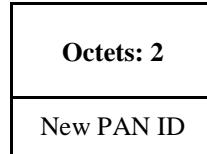
3.4.10.3.4.1 PAN Identifier Update

7645
7646

If the value of the Update Command Identifier sub-field indicates a PAN identifier update, then the Update Information field shall have the format shown in Figure 3.31.

7647

Figure 3.31 PAN Identifier Update



7648
7649
7650

The PAN identifier update shall be made up of a single 16-bit PAN identifier that is the new PAN identifier for this network to use. The Update Information count sub field shall be set equal to 1 as there is only a single PAN identifier contained within the Update Information field.

7651

3.4.11 End Device Timeout Request Command

7652
7653
7654

The End Device Timeout Request command is sent by an end device informing its parent of its timeout requirements. This allows the parent the ability to delete the child entry from the neighbor table if the child has not communicated with the parent in the specified amount of time.

7655

The payload of an End Device Timeout Request command shall be formatted as illustrated in Figure 3.32.

7656

7657

Figure 3.32 Format of the End Device Timeout Request Command

Octets: 1	1
Request Timeout Enumeration	End Device Configuration

7658

3.4.11.1 MAC Data Service Requirements

In order to transmit this command using the MAC data service, specified in [B1], the following information shall be provided:

- The destination address and PAN identifier shall be set to the network address and PAN identifier, respectively, of the end device's parent.
- The source MAC address and PAN identifier shall be set to the network address and PAN identifier of the device transmitting the End Device Timeout Request command.
- The transmission options shall be set to require acknowledgement.
- The address mode and intra-PAN flags shall be set to support the addressing fields described here.

3.4.11.2 NWK Header fields

The NWK header fields of the End Device Timeout Request command frame shall be set as follows:

- The source address field of the NWK header shall be set to the 16-bit network address.
- The source IEEE address sub-field of the frame control field shall be set to 1, and the source IEEE address field shall be set to the IEEE address of the device issuing the request.
- The destination address field in the NWK header shall be set to the 16-bit network address of the parent.
- The destination IEEE address sub-field of the frame control field shall be set to 1, and the destination IEEE address field shall be set to the IEEE address of the parent.
- The radius field shall be set to 1.

3.4.11.3 NWK Payload Fields

The NWK frame payload contains a command identifier field and a capability

Table 3.43 Fields of the End Device Timeout Request

Name	Type	Valid Range	Description
Requested Timeout Enumeration	Enumerated type	0 - 14	The requested timeout enumeration. This will be converted into actual timeout value based on Table 3.44
End Device Configuration	Bitmask	0x00 – 0x00	This is an enumeration of the child's requested configuration.

3.4.11.3.1 Requested Timeout Field

The valid values for the requested timeout will be an enumerated type between 0 and 14. This will be converted to an actual timeout value according to Table 3.44 below

Table 3.44 Requested Timeout Enumerated Values

Requested Timeout Enumeration Value	Actual Timeout Value
0	10 seconds
1	2 minutes

2	4 minutes
3	8 minutes
4	16 minutes
5	32 minutes
6	64 minutes
7	128 minutes
8	256 minutes
9	512 minutes
10	1024 minutes
11	2048 minutes
12	4096 minutes
13	8192 minutes
14	16384 minutes

7687 This allows for an actual timeout value between 10 seconds and 16384 minutes (~ 11 days).

7688 **3.4.11.3.2 End Device Configuration Field**

7689 This is a bitmask indicating the end device's requested configuration. At this time there are no enumerat-
7690 ed bits in the configuration field. Devices adhering to this standard shall set the field to 0. To allow for
7691 future compatibility this field is left in place. Devices that receive the End Device Timeout Request mes-
7692 sage with an End Device Configuration field set to anything other than 0 shall reject the message. The
7693 receiving device shall send an End Device Timeout Response command with a status of 0x01 (INCOR-
7694 RECT_VALUE).

7695

7696 **3.4.12 End Device Timeout Response Command**

7697 The End Device Timeout Response is sent by a router parent informing the end device whether it has ac-
7698 cepted the timeout value that it was previously sent, and what its capabilities are.

7699 **Figure 3.33 Format of the End Device Timeout Response Command**

Octets: 1	1
Status	Parent Information

7700

7701 **3.4.12.1 MAC Data Service Requirements**

7702 In order to transmit this command using the MAC data service, specified in reference [B1], the following
7703 information shall be provided:

- 7704
- 7705 • The destination address and PAN identifier shall be set to the network address and PAN identifier,
respectively, of the end device.

7706

 - 7707 • The source MAC address and PAN identifier shall be set to the network address and PAN identi-
fier of the device transmitting the End Device Timeout Response command.

7708

 - The transmission options shall be set to require acknowledgement.

- 7709 • The address mode and intra-PAN flags shall be set to support the addressing fields described here.

3.4.12.2 NWK Header fields

7711 The NWK header fields of the End Device Timeout Response command frame shall be set as follows:

- 7712 • The source address field of the NWK header shall be set to the 16-bit network address.
- 7713 • The source IEEE address sub-field of the frame control field shall be set to 1, and the source IEEE
7714 address field shall be set to the IEEE address of the device issuing the command.
- 7715 • The destination address field in the NWK header shall be set to the 16-bit network address of the
7716 end device.
- 7717 • The destination IEEE address sub-field of the frame control field shall be set to 1, and the destina-
7718 tion IEEE address field shall be set to the IEEE address of the end device.
- 7719 • The radius field shall be set to 1.

3.4.12.2.1 NWK Payload Fields

7721 The NWK frame payload contains a command identifier field and a capability information field. The
7722 payload of the End Device Timeout Response are described in Table 3.45.

7724 **Table 3.45 Payload fields of the End Device Timeout Response**

Name	Type	Valid Range	Description
Status	Enumeration	0 – 0xFF	The success or failure result of the previously received End Device Timeout Request command. See Table 3.46 for an enumeration of the status codes.
Parent Information	Bitmask	0 – 0xFF	This bitmask indicates the parent router's support information to the child device. The bitmask's values are described in Table 3.47

7725 **Table 3.46 Enumeration of the End Device Timeout Response Status**

Status	Value	Description
SUCCESS	0x00	The End Device Timeout Request message was accepted by the parent.
INCORRECT_VALUE	0x01	The received timeout value in the End Device Timeout Request command was outside the allowed range.
Reserved	0x02 – 0xFF	Reserved for Future Use

7728

Table 3.47 Values of the Parent Information Bitmask

Bits	Description
0	MAC Data Poll Keepalive Supported
1	End Device Timeout Request Keepalive Supported
2 – 15	Reserved for future use

7729

7730 **3.5 Constants and NIB Attributes**

7731 **3.5.1 NWK Constants**

7732 The constants that define the characteristics of the NWK layer are presented in Table 3.48.

7733 **Table 3.48 NWK Layer Constants**

Constant	Description	Value
<i>nwkcCoordinatorCapable</i>	A Boolean flag indicating whether the device is capable of becoming the ZigBee coordinator. A value of 0x00 indicates that the device is not capable of becoming a coordinator while a value of 0x01 indicates that the device is capable of becoming a coordinator.	Configuration dependent
<i>nwkcDefaultSecurityLevel</i>	The default security level to be used (see Chapter 4).	Defined in stack profile
<i>nwkcMinHeaderOverhead</i>	The minimum number of octets added by the NWK layer to an NSDU.	0x08
<i>nwkcProtocolVersion</i>	The version of the ZigBee NWK protocol in the device.	0x02
<i>nwkcWaitBeforeValidation</i>	The number of OctetDurations, on the originator of a multicast route request, between receiving a route reply and sending a message to validate the route.	0x9c40 (0x500 msec on 2.4 GHz)
<i>nwkcRouteDiscoveryTime</i>	The number of OctetDurations until a route discovery expires.	0x4c4b4 (0x2710 msec on 2.4GHz)
<i>nwkcMaxBroadcastJitter</i>	The maximum broadcast jitter time measured in OctetDurations.	0x7d0 (0x40 msec on 2.4GHz)

Constant	Description	Value
<i>nwkInitialRREQRetries</i>	The number of times the first broadcast transmission of a route request command frame is retried.	0x03
<i>nwkRREQRetries</i>	The number of times the broadcast transmission of a route request command frame is retried on relay by an intermediate ZigBee router or ZigBee coordinator.	0x02
<i>nwkRREQRetryInterval</i>	The number of OctetDurations between retries of a broadcast route request command frame.	0x1f02 (0xfe msec on 2.4Ghz)
<i>nwkMinRREQJitter</i>	The minimum jitter, in OctetDurations, for broadcast retransmission of a route request command frame.	0x3f (2 msec on 2.4GHz)
<i>nwkMaxRREQJitter</i>	The maximum jitter, in OctetDurations, for broadcast retransmission of a route request command frame.	0xfa0 (128 msec on 2.4GHz)
<i>nwkMACFrameOverhead</i>	The size of the MAC header used by the ZigBee NWK layer.	0x0b

7734

3.5.2 NWK Information Base

7735
7736
7737
7738
7739
7740
7741

The NWK information base (NIB) comprises the attributes required to manage the NWK layer of a device. Each of these attributes can be read or written using the NLME-GET.request and NLME-SET.request primitives, respectively, except for attributes for which the Read Only column contains a value of Yes. In that case, the attributes value may be read using the NLME-GET.request primitive but may not be set using the NLME-SET.request primitive. Generally, these read-only attribute are set using some other mechanism. For example, the *nwkSequenceNumber* attribute is set as specified in section 3.6.2.1 and incremented every time the NWK layer sends a frame. The attributes of the NIB are presented in Table 3.49.

7742

Table 3.49 NIB Attributes

Attribute	Id	Type	Read Only	Range	Description	Default
<i>nwkSequenceNumber</i>	0x81	Integer	Yes	0x00 – 0xff	A sequence number used to identify outgoing frames (see section 3.6.2).	Random value from within the range
<i>nwkPassiveAckTimeout</i>	0x82	Integer	No	0x000000 – 0xffffffff	The maximum time duration in OctetDurations allowed for the parent and all child devices to retransmit a broadcast message (passive ac-	Defined in stack profile

Attribute	Id	Type	Read Only	Range	Description	Default
					knowledgment time-out).	
<i>nwkMaxBroadcastRetries</i>	0x83	Integer	No	0x00 – 0x5	The maximum number of retries allowed after a broadcast transmission failure.	0x03
<i>nwkMaxChildren</i>	0x84	Integer	No	0x00 – 0xff	The number of children a device is allowed to have on its current network. Note that when <i>nwkAddrAlloc</i> has a value of 0x02, indicating stochastic addressing, the value of this attribute is implementation-dependent.	Defined in the stack profile
<i>nwkMaxDepth</i>	0x85	Integer	Yes	0x00 – 0xff	The depth a device can have.	Defined in stack profile
<i>nwkMaxRouters</i>	0x86	Integer	No	0x01-0xff	The number of routers any one device is allowed to have as children. This value is determined by the ZigBee coordinator for all devices in the network. If <i>nwkAddrAlloc</i> is 0x02 this value not used.	Defined in stack profile
<i>nwkNeighborTable</i>	0x87	Set	No	Variable	The current set of neighbor table entries in the device (see Table 3.53).	Null set

Attribute	Id	Type	Read Only	Range	Description	Default
<i>nwkNetworkBroadcastDeliveryTime</i>	0x88	Integer	No	0 – 0xffffffff	Time duration in OctetDurations that a broadcast message needs to encompass the entire network. This is a calculated quantity based on other NIB attributes. The formula is given in section 3.5.2.1.	Defined in stack profile
<i>nwkReportConstantCost</i>	0x89	Integer	No	0x00-0x01	If this is set to 0, the NWK layer shall calculate link cost from all neighbor nodes using the LQI values reported by the MAC layer; otherwise, it shall report a constant value.	0x00
<i>Reserved</i>	0x8a					
<i>nwkRouteTable</i>	0x8b	Set	No	Variable	The current set of routing table entries in the device (see Table 3.56).	Null set
<i>nwkSymLink</i>	0x8e	Boolean	No	TRUE or FALSE	The current route symmetry setting: TRUE means that routes are considered to be comprised of symmetric links. Backward and forward routes are created during one-route discovery and they are identical. FALSE indicates that routes are not considered to be comprised of symmetric links. Only the forward route is stored during route discovery.	FALSE
<i>nwkCapabilityInformation</i>	0x8f	Bit vector	Yes	See Table 3.52.	This field shall contain the device capability information established at network joining time.	0x00

Attribute	Id	Type	Read Only	Range	Description	Default
<i>nwkAddrAlloc</i>	0x90	Integer	No	0x00 - 0x02	A value that determines the method used to assign addresses: 0x00 = use distributed address allocation 0x01 = reserved 0x02 = use stochastic address allocation	0x00
<i>nwkUseTreeRouting</i>	0x91	Boolean	No	TRUE or FALSE	A flag that determines whether the NWK layer should assume the ability to use hierarchical routing: TRUE = assume the ability to use hierarchical routing. FALSE = never use hierarchical routing.	TRUE
<i>nwkManagerAddr</i>	0x92	Integer	No	0x0000 - 0xffff	The address of the designated network channel manager function.	0x0000
<i>nwkMaxSourceRoute</i>	0x93	Integer	No	0x00 - 0xff	The maximum number of hops in a source route.	0x0c
<i>nwkUpdateId</i>	0x94	Integer	No	0x00 - 0xFF	The value identifying a snapshot of the network settings with which this node is operating with.	0x00
<i>nwkTransactionPersistenceTime</i>	0x95	Integer	No	0x0000 - 0xffff	The maximum time (in superframe periods) that a transaction is stored by a coordinator and indicated in its beacon. This attribute reflects the value of the MAC PIB attribute <i>macTransactionPersistenceTime</i> (see [B1]) and any changes made by the higher layer will be reflected in the MAC PIB attribute value as well.	0x01f4

Attribute	Id	Type	Read Only	Range	Description	Default
<i>nwkNetworkAddress</i>	0x96	Integer	No	0x0000 - 0xffff7	The 16-bit address that the device uses to communicate with the PAN. This attribute reflects the value of the MAC PIB attribute <i>mac-ShortAddress</i> (see [B1]) and any changes made by the higher layer will be reflected in the MAC PIB attribute value as well.	0xffff
<i>nwkStackProfile</i>	0x97	Integer	No	0x00-0x0f	The identifier of the ZigBee stack profile in use for this device.	
<i>nwkBroadcastTransactionTable</i>	0x98	Set	Yes	-	The current set of broadcast transaction table entries in the device (see Table 3.60).	Null set
<i>nwkGroupIDTable</i>	0x99	Set	No	Variable	The set of group identifiers, in the range 0x0000 - 0xffff, for groups of which this device is a member.	Null Set
<i>nwkExtendedPANID</i>	0x9a	64-bit extended address	No	0x00000000 00000000 - 0xffffffffffff ffe	The Extended PAN Identifier for the PAN of which the device is a member. The value 0x0000000000000000 means the Extended PAN Identifier is unknown.	0x00000000 00000000
<i>nwkUseMulticast</i>	0x9b	Boolean	No	TRUE or FALSE	A flag determining the layer where multicast messaging occurs. TRUE = multicast occurs at the network layer. FALSE= multicast occurs at the APS layer and using the APS header.	TRUE
<i>nwkRouteRecordTable</i>	0x9c	Set	No	Variable	The route record table (see Table 3.50).	Null Set

Attribute	Id	Type	Read Only	Range	Description	Default
<i>nwkIsConcentrator</i>	0x9d	Boolean	No	TRUE or FALSE	A flag determining if this device is a concentrator. TRUE = Device is a concentrator. FALSE = Device is not a concentrator.	FALSE
<i>nwkConcentratorRadius</i>	0x9e	Integer	No	0x00 - 0xff	The hop count radius for concentrator route discoveries.	0x0000
<i>nwkConcentratorDiscoveryTime</i>	0x9f	Integer	No	0x00 - 0xff	The time in seconds between concentrator route discoveries. If set to 0x0000, the discoveries are done at start up and by the next higher layer only.	0x0000
<i>nwkSecurityLevel</i>	0xa0		No		Security attribute defined in Chapter 4.	
<i>nwkSecurityMaterialSet</i>	0xa1		No		Security attribute defined in Chapter 4.	
<i>nwkActiveKeySeqNumber</i>	0xa2		No		Security attribute defined in Chapter 4.	
<i>nwkAllFresh</i>	0xa3		No		Security attribute defined in Chapter 4.	
<i>nwkLinkStatusPeriod</i>	0xa6	Integer	No	0x00 - 0xff	The time in seconds between link status command frames.	0x0f
<i>nwkRouterAgeLimit</i>	0xa7	Integer	No	0x00 - 0xff	The number of missed link status command frames before resetting the link costs to zero.	3

Attribute	Id	Type	Read Only	Range	Description	Default
<i>nwkUniqueAddr</i>	0xa8	Boolean	No	TRUE or FALSE	A flag that determines whether the NWK layer should detect and correct conflicting addresses: TRUE = assume addresses are unique. FALSE = addresses may not be unique.	TRUE
<i>nwkAddressMap</i>	0xa9	Set	No	Variable	The current set of 64-bit IEEE to 16-bit network address map (see Table 3.51).	Null Set
<i>nwkTimeStamp</i>	0x8C	Boolean	No	TRUE or FALSE	A flag that determines if a time stamp indication is provided on incoming and outgoing packets. TRUE= time indication provided. FALSE = no time indication provided.	FALSE
<i>nwkPANId</i>	0x80	16-bit PAN ID	No	0x0000 - 0xffff	This NIB attribute should, at all times, have the same value as <i>macPANId</i> .	0xffff
<i>nwkTxTotal</i>	0x8D	Integer	No	0x0000 - 0xffff	A count of unicast transmissions made by the NWK layer on this device. Each time the NWK layer transmits a unicast frame, by invoking the MCPS-DATA.request primitive of the MAC sub-layer, it shall increment this counter. When either the NHL performs an NLME-SET.request on this attribute or if the value of <i>nwkTxTotal</i> rolls over past 0xffff the NWK layer shall reset to 0x00 each Transmit Failure field contained in the neighbor table.	0

Attribute	Id	Type	Read Only	Range	Description	Default
<i>nwkLeaveRequestAllowed</i>	0xAA	Boolean	No	TRUE or FALSE	This policy determines whether or not a remote NWK leave request command frame received by the local device is accepted.	TRUE
<i>nwkParentInformation</i>	0xAB	Bitmask	No	0x00 – 0xFF	The behavior depends upon whether the device is an FFD or RFD. For an RFD, this records the information received in an End Device Timeout Response command indicating the parent information. The bitmask values are defined in Table 3.47. For an FFD, this records the device's local capabilities.	0x00
<i>nwkEndDeviceTimeoutDefault</i>	0xAC	Integer	No	0x00 – 0xFF	This is an index into tableTable 3.44. It indicates the default timeout in minutes for any end device that does not negotiate a different timeout value.	8
<i>nwkLeaveRequestWithoutRejoinAllowed</i>	0xAD	Boolean	No	TRUE or FALSE	This policy determines whether a NWK leave request is accepted when the Rejoin bit in the message is set to FALSE	TRUE
<i>nwkIeeeAddress</i>	0xAE	64-bit address	Yes	0x00000000 00000001 – 0xFFFFFFFF FFFFFFFF	The IEEE address of the local device.	

7744

Table 3.50 Route Record Table Entry Format

Field Name	Field Type	Valid Range	Reference
Network Address	Integer	0x0000- 0xffff	The destination network address for this route record.
Relay Count	Integer	0x0000 - 0xffff	The count of relay nodes from concentrator to the destination.
Path	Set of Network Addresses		The set of network addresses that represent the route in order from the concentrator to the destination.

7745

7746

Table 3.51 Network Address Map

64-bit IEEE Address	16-bit Network address
A valid 64-bit IEEE Address or Null if not known	0x0000 - 0xffff

7747

3.5.2.1 Broadcast Delivery Time

7748
7749

The total delivery time for a broadcast transmission, *i.e.* the time required for a broadcast to be delivered to every device in the network, shall be calculated according to the following formula:

7750
7751
7752
7753

```
nwkBroadcastDeliveryTime = 2*nwkMaxDepth*
                           ((0.05+(nwkMaxBroadcastJitter/2))+
                            nwkPassiveAckTimeout*nwkBroadcastRetries/
                            1000)
```

7754

3.6 Functional Description

7755

3.6.1 Network and Device Maintenance

7756

All ZigBee devices shall provide the following functionality:

7757
7758
7759

- Join a network
- Leave a network
- Rejoin a network

7760

Both ZigBee coordinators and routers shall provide the following additional functionality:

7761
7762

- Permit devices to join the network using the following:
 - Association indications from the MAC

- 7763 • Explicit join requests from the application
7764 • Rejoin requests
7765 • Permit devices to leave the network using the following:
7766 • Network leave command frames
7767 • Explicit leave requests from the application
7768 • Participate in assignment of logical network addresses
7769 • Maintain a list of neighboring devices

7770 ZigBee coordinators shall provide functionality to establish a new network. ZigBee routers and end devices
7771 shall provide the support of portability within a network.

3.6.1.1 Establishing a New Network

7773 The procedure to establish a new network is initiated through use of the
7774 NLME-NETWORK-FORMATION.request primitive. Only devices for which the *nwkCoordinatorCapable*
7775 constant has a value of 0x01, and which are not currently joined to a network shall attempt to establish a
7776 new network. If this procedure is initiated on any other device, the NLME shall terminate the procedure
7777 and notify the next higher layer of the illegal request. This is achieved by issuing the
7778 NLME-NETWORK-FORMATION.confirm primitive with the Status parameter set to INVALID_REQUEST.
7779

7780 When this procedure is initiated, the NLME shall first request that the MAC sub-layer perform an energy
7781 detection scan over either a specified set of channels or, by default, the complete set of available channels,
7782 as dictated by the PHY layer (see [B1]), to search for possible interferers. A channel scan is initiated by is-
7783 suing the MLME-SCAN.request primitive to the MAC sub-layer with the ScanType parameter set to ener-
7784 gy detection scan. The results are communicated back via the MLME-SCAN.confirm primitive. This scan
7785 is not necessary if there is only one channel specified.

7786 On receipt of the results from a successful energy detection scan, the NLME shall order the channels ac-
7787 cording to increasing energy measurement and discard those channels whose energy levels are beyond an
7788 acceptable level. The choice of an acceptable energy level is left to the implementation. The NLME shall
7789 then perform an active scan, by issuing the MLME-SCAN.request primitive with the ScanType parameter
7790 set to active scan and ChannelList set to the list of acceptable channels and ChannelPage set to zero, to
7791 search for other ZigBee devices. To determine the best channel on which to establish a new network, the
7792 NLME shall review the list of returned PAN descriptors and find the first channel with the lowest number
7793 of existing networks, favoring a channel with no detected networks.

7794 If no suitable channel is found, the NLME shall terminate the procedure and notify the next higher layer of
7795 the startup failure. This is achieved by issuing the NLME-NETWORK-FORMATION.confirm primitive
7796 with the Status parameter set to STARTUP_FAILURE.

7797 If a suitable channel is found, the NLME shall select a PAN identifier for the new network. To do this the
7798 device shall choose a random PAN identifier less than 0xffff that is not already in use on the selected
7799 channel. Once the NLME makes its choice, it shall set the *macPANID* attribute in the MAC sub-layer to
7800 this value by issuing the MLME-SET.request primitive.

7801 If no unique PAN identifier can be chosen, the NLME shall terminate the procedure and notify the next
7802 higher layer of the startup failure by issuing the NLME-NETWORK-FORMATION.confirm primitive with
7803 the Status parameter set to STARTUP_FAILURE.

7804 Once a PAN identifier is selected, the NLME shall select a 16-bit network address equal to 0x0000 and set
7805 the *nwkNetworkAddress* attribute of the NIB equal to the selected network address.

7806 Once a network address is selected, the NLME shall check the value of the *nwkExtendedPANId* attribute of
7807 the NIB. If this value is 0x0000000000000000 this attribute is initialized with the value of the MAC con-
7808 stant *aExtendedAddress*.

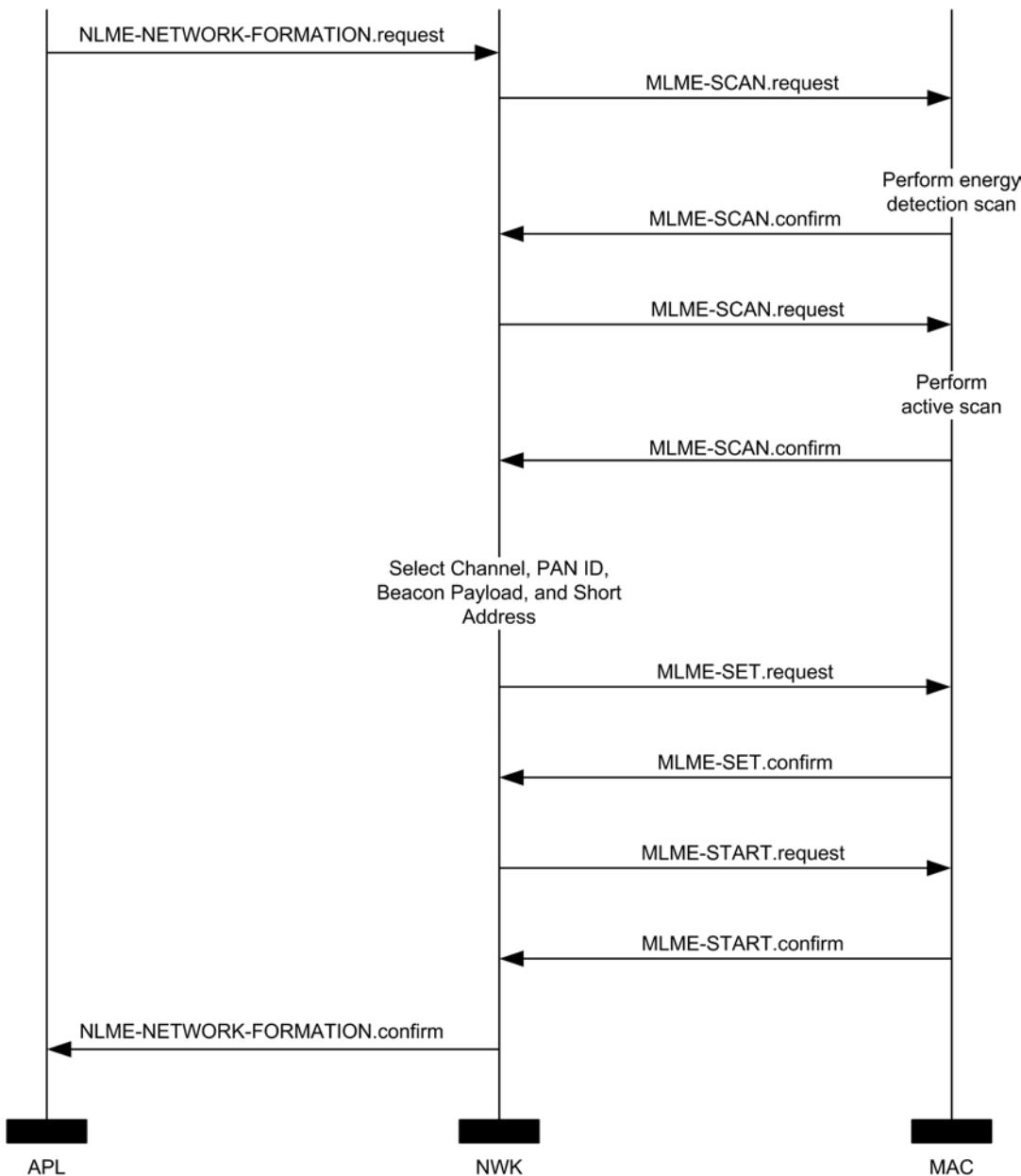
7809 Once the value of the *nwkExtendedPANId* is checked, the NLME shall begin operation of the new PAN by
 7810 issuing the MLME-START.request primitive to the MAC sub-layer. The parameters of the
 7811 MLME-START.request primitive shall be set according to those passed in the
 7812 NLME-NETWORK-FORMATION.request, the results of the channel scan, and the chosen PAN identifier.
 7813 The status of the PAN startup is communicated back via the MLME-START.confirm primitive.

7814 On receipt of the status of the PAN startup, the NLME shall inform the next higher layer of the status of its
 7815 request to initialize the ZigBee coordinator. This is achieved by issuing the
 7816 NLME-NETWORK-FORMATION.confirm primitive with the Status parameter set to the primitive re-
 7817 turned in the MLME-START.confirm from the MAC sub-layer.

7818 The procedure to successfully start a new network is illustrated in the message sequence chart (MSC)
 7819 shown in Figure 3.34.

7820

Figure 3.34 Establishing a New Network



7821

3.6.1.2 Permitting Devices to Join a Network

The procedure for permitting devices to join a network is initiated through the NLME-PERMIT-JOINING.request primitive. Only devices that are either the ZigBee coordinator or a ZigBee router shall attempt to permit devices to join the network.

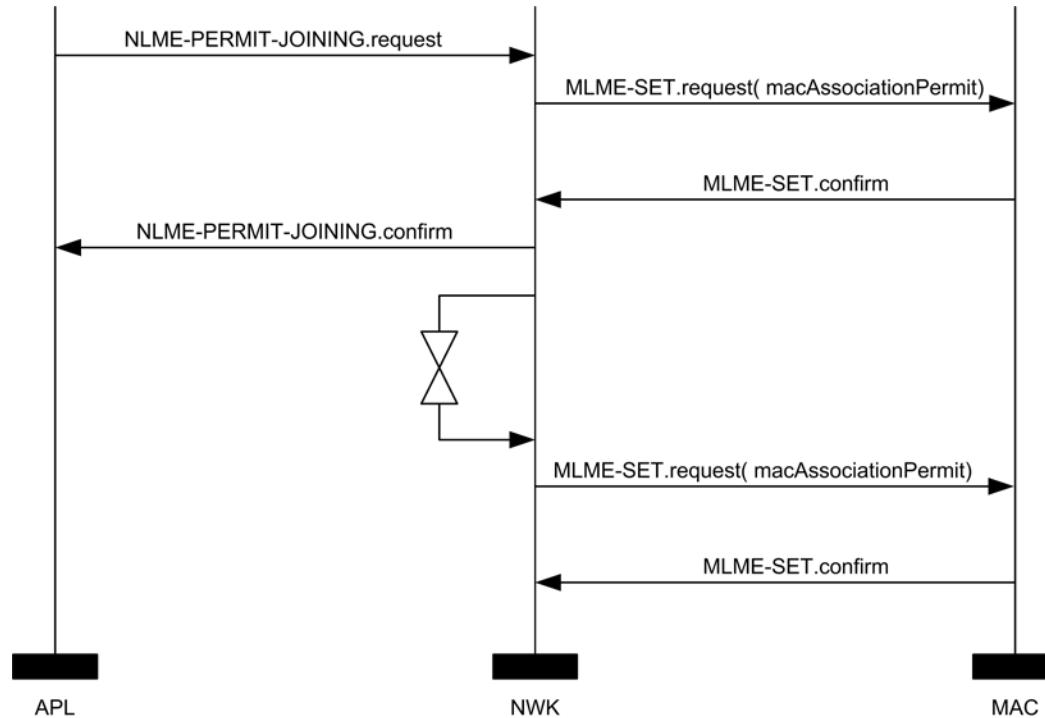
When this procedure is initiated with the PermitDuration parameter set to 0x00, the NLME shall set the *macAssociationPermit* PIB attribute in the MAC sub-layer to FALSE. A MAC sub-layer attribute setting is initiated by issuing the MLME-SET.request primitive.

When this procedure is initiated with the PermitDuration parameter set to a value between 0x01 and 0xfe, the NLME shall set the *macAssociationPermit* PIB attribute in the MAC sub-layer to TRUE. The NLME shall then start a timer to expire after the specified duration. On expiration of this timer, the NLME shall set the *macAssociationPermit* PIB attribute in the MAC sub-layer to FALSE.

When this procedure is initiated with the PermitDuration parameter set to 0xff, the NLME shall set the *macAssociationPermit* PIB attribute in the MAC sub-layer to TRUE for an unlimited amount of time, unless another NLME-PERMIT-JOINING.request primitive is issued.

The procedure for permitting devices to join a network is illustrated in the MSC shown in Figure 3.35.

Figure 3.35 Permitting Devices to Join a Network



3.6.1.3 Network Discovery

The NWK layer enables higher layers to discover what networks, if any, are operational in the POS of a device.

The procedure for network discovery shall be initiated by issuing the NLME-NETWORK-DISCOVERY.request primitive with the ScanChannels parameter set to indicate which channels are to be scanned for networks and the ScanDuration parameter set to indicate the length of time to be spent scanning each channel. Upon receipt of this primitive, the NWK layer shall issue an MLME-SCAN.request primitive asking the MAC sub-layer to perform an active scan.

7847 Every beacon frame received during the scan having a non-zero length payload shall cause the
7848 MLME-BEACON-NOTIFY.indication primitive to be issued from the MAC sub-layer of the scanning de-
7849 vice to its NLME. This primitive includes information such as the addressing information of the beaconing
7850 device, whether or not it is permitting association and the beacon payload. (See [B1] for the complete list of
7851 parameters). The NLME of the scanning device shall check the protocol ID field in the beacon payload and
7852 verify that it matches the ZigBee protocol identifier. If not, the beacon is ignored. Otherwise, the device
7853 shall copy the relevant information from each received beacon (see Figure 3.51 for the structure of the
7854 beacon payload) into its neighbor table (see Table 3.53 for the contents of a neighbor table entry).

7855 Once the MAC sub-layer signals the completion of the scan by issuing the MLME-SCAN.confirm primitive
7856 to the NLME, the NWK layer shall issue the NLME-NETWORK-DISCOVERY.confirm primitive
7857 containing a description of each network that was heard. Every network description contains the ZigBee
7858 version, stack profile, Extended PAN Id, PAN Id, logical channel, and information on whether it is permit-
7859 ting joining (see Table 3.8).

7860 **3.6.1.4 Joining a Network**

7861 For purposes of the ensuing discussion, a parent-child relationship is formed when a device having mem-
7862 bership in the network allows a new device to join. On joining, the new device becomes the child, while the
7863 first device becomes the parent.

7864 **3.6.1.4.1 Joining a Network Through Association**

7865 This section specifies the procedure a device (child) shall follow if it opts to join a network using the un-
7866 derlying association capabilities provided by the MAC, as well as the procedure a ZigBee coordinator or
7867 router (parent) shall follow upon receipt of an MLME-ASSOCIATE.request primitive from the MAC.

7868 **3.6.1.4.1.1 Child Procedure**

7869 The procedure for joining a network using the MAC layer association procedure should be preceded by
7870 network discovery as described in section 3.6.1.3. Upon receipt of the
7871 NLME-NETWORK-DISCOVERY.confirm primitive, the next higher layer shall either choose a network
7872 to join from the discovered networks or redo the network discovery. Once a network is selected, it shall
7873 then issue the NLME-JOIN.request with the RejoinNetwork parameter set to 0x00 and the JoinAsRouter
7874 parameter set to indicate whether the device wants to join as a routing device.

7875 Only those devices that are not already joined to a network shall initiate the join procedure. If any other de-
7876 vice initiates this procedure, the NLME shall terminate the procedure and notify the next higher layer of the
7877 illegal request by issuing the NLME-JOIN.confirm primitive with the Status parameter set to INVA-
7878 LID_REQUEST.

7879 For a device that is not already joined to a network, the NLME-JOIN.request primitive shall cause the
7880 NWK layer to search its neighbor table for a suitable parent device, *i.e.* a device for which following condi-
7881 tions are true:

- 7882 • The device belongs to the network identified by the ExtendedPANId parameter.
- 7883 • The device is open to join requests and is advertising capacity of the correct device type.
- 7884 • The link quality for frames received from this device is such that a link cost of at most 3 is produced
7885 when calculated as described in section 3.6.3.1.
- 7886 • If the neighbor table entry contains a potential parent field for this device, that field shall have a value
7887 of 1 indicating that the device is a potential parent.
- 7888 • The device shall have the most recent update id, where the determination of most recent needs to take
7889 into account that the update id will wrap back to zero. In particular the update id given in the beacon
7890 payload of the device should be greater than or equal to — again, compensating for wrap — the
7891 *nwkUpdateId* attribute of the NIB.

7892 If the neighbor table contains no devices that are suitable parents, the NLME shall respond with an
 7893 NLME-JOIN.confirm with a Status parameter of NOT_PERMITTED. If the neighbor table has more than
 7894 one device that could be a suitable parent, the device which is at a minimum depth from the ZigBee coor-
 7895 dinator may be chosen if and only if the *nwkStackProfile* is set to 1. If more than one device has a mini-
 7896 mum depth, the NWK layer is free to choose from among them. If *nwkStackProfile* is not equal to 1, then
 7897 the depth shall not be considered when choosing a suitable parent.

7898 Once a suitable parent is identified the device shall set its *nwkParentInformation* value in the NIB to 0, then
 7899 the NLME shall issue an MLME-ASSOCIATE.request primitive to the MAC sub-layer. The LogicalChan-
 7900 nel parameter of the MLME-ASSOCIATE.request primitive shall be set to that found in the neighbor table
 7901 entry corresponding to the coordinator address of the potential parent. The bit-fields of the CapabilityIn-
 7902 formation parameter shall have the values shown in Table 3.52 and the capability information shall be
 7903 stored as the value of the *nwkCapabilityInformation* NIB attribute (see Table 3.49). If more than one device
 7904 meets these requirements, then the joining device may select the parent with the smallest network depth.

7905

Table 3.52 Capability Information Bit-Fields

Bit	Name	Description
0	Alternate PAN coordinator	This field will always have a value of 0 in implementations of this specification.
1	Device type	This field will have a value of 1 if the joining device is a ZigBee router. It will have a value of 0 if the device is a ZigBee end device or else a router-capable device that is joining as an end device.
2	Power source	This field will be set to the value of lowest-order bit of the PowerSource parameter passed to the NLME-JOIN-request primitive. The values are: 0x01 = Mains-powered device 0x00 = other power source
3	Receiver on when idle	This field will be set to the value of the lowest-order bit of the RxOnWhenIdle parameter passed to the NLME-JOIN.request primitive. 0x01 = The receiver is enabled when the device is idle 0x00 = The receiver may be disabled when the device is idle
4 – 5	Reserved	This field will always have a value of 0 in implementations of this specification.
6	Security capability	This field shall have a value of 0. Note that this overrides the default meaning specified in [B1].

Bit	Name	Description
7	Allocate address	This field will have a value of 1 in implementations of this specification, indicating that the joining device must be issued a 16-bit network address, except in the case where a device has self-selected its address while using the NWK rejoin command to join a network for the first time in a secure manner. In this case, it shall have a value of 0.

7906

7907 Otherwise, the NLME issues the NLME-JOIN.confirm with the Status parameter set to the Status parameter value returned from the MLME-ASSOCIATE.confirm primitive.
7908

7909 If the RejoinNetwork parameter is 0x00 and the JoinAsRouter parameter is set to TRUE, the device will
7910 function as a ZigBee router in the network. If the JoinAsRouter parameter is FALSE, then it will join as an
7911 end device and not participate in routing.

7912 The addressing parameters in the MLME-ASSOCIATE.request primitive (see Chapter 2) shall be set to
7913 contain the addressing information for the device chosen from the neighbor table. The status of the associa-
7914 tion is communicated back to the NLME via the MLME-ASSOCIATE.confirm primitive.

7915 If the attempt to join was unsuccessful, the NWK layer shall receive an MLME-ASSOCIATE.confirm
7916 primitive from the MAC sub-layer with the Status parameter indicating the error. If the Status parameter
7917 indicates a refusal to permit joining on the part of the neighboring device (that is, PAN at capacity or PAN
7918 access denied), then the device attempting to join should set the Potential parent bit to 0 in the correspond-
7919 ing neighbor table entry to indicate a failed join attempt. Setting the Potential parent bit to 0 ensures that
7920 the NWK layer shall not issue another request to associate to the same neighboring device. The Potential
7921 parent bit should be set to 1 for every entry in the neighbor table each time an MLME-SCAN.request primi-
7922 tive is issued.

7923 A join request may also be unsuccessful, if the potential parent is not allowing new routers to associate (for
7924 example, the maximum number of routers, *nwkMaxRouters* may already have associated with the device)
7925 and the joining device has set the JoinAsRouter parameter to TRUE. In this case, the NLME-JOIN.confirm
7926 primitive will indicate a status of NOT_PERMITTED. In this case, the child device's application may wish
7927 to attempt to join again as an end device instead, by issuing another NLME-JOIN.request with the
7928 JoinAsRouter parameter set to FALSE.

7929 If the attempt to join was unsuccessful, the NLME shall attempt to find another suitable parent from the
7930 neighbor table. If no such device could be found, the NLME shall issue the NLME-JOIN.confirm primitive
7931 with the Status parameter set to the value returned in the MLME-ASSOCIATE.confirm primitive.

7932 If the attempt to join was unsuccessful and there is a second neighboring device that could be a suitable
7933 parent, the NWK layer shall initiate the MAC sub-layer association procedure with the second device. The
7934 NWK layer shall repeat this procedure until it either joins the PAN successfully or exhausts its options to
7935 join the PAN.

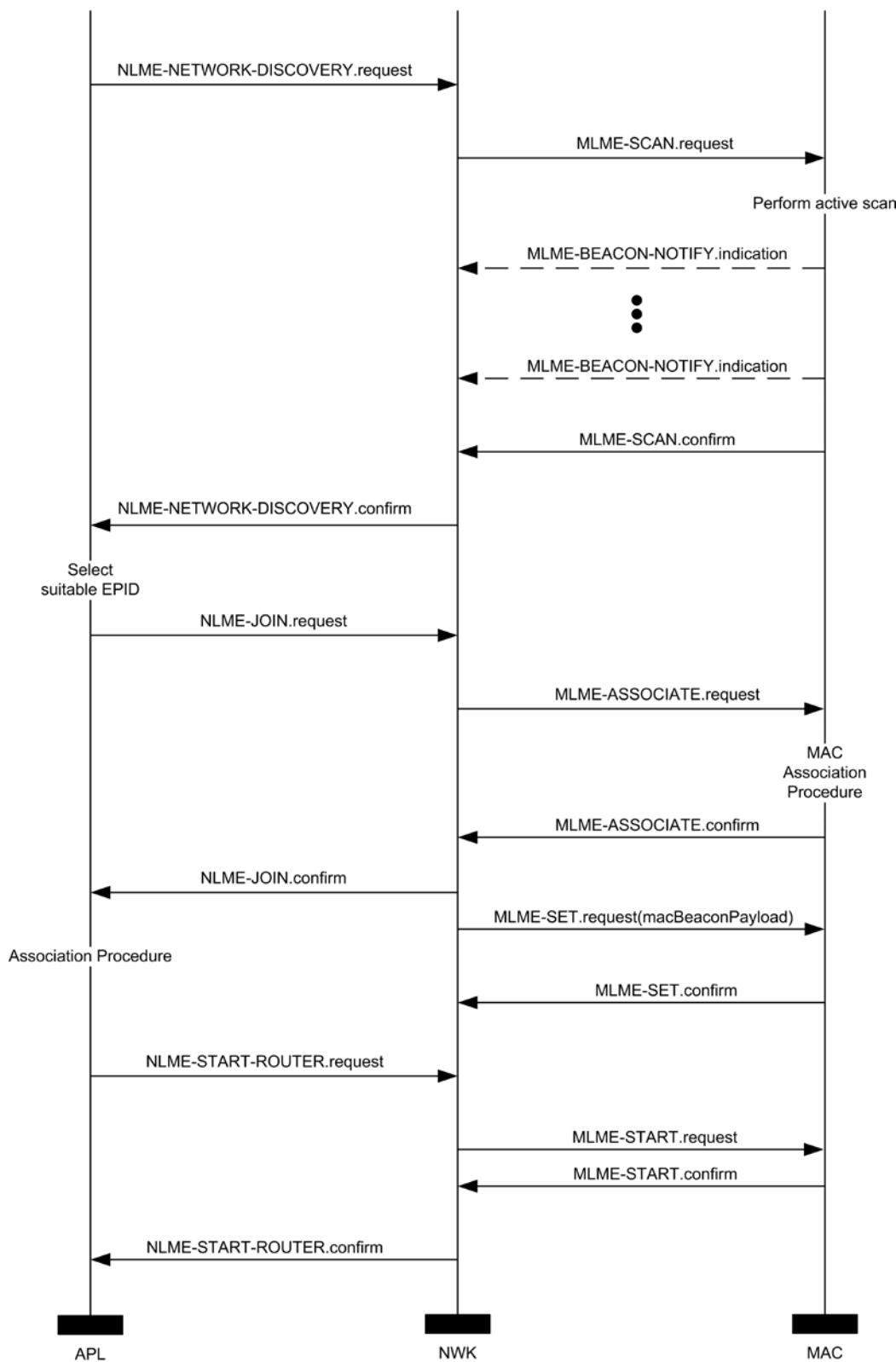
7936 If the device cannot successfully join the PAN specified by the next higher layer, the NLME shall terminate
7937 the procedure by issuing the NLME-JOIN.confirm primitive with the Status parameter set to the value re-
7938 turned in the last received MLME-ASSOCIATE.confirm primitive. In this case, the device shall not receive
7939 a valid logical address and shall not be permitted to transmit on the network.

7940 If the attempt to join was successful, the NWK shall issue the NLME-JOIN.confirm primitive with a status
7941 value of SUCCESS. In this case, the MLME-ASSOCIATE.confirm primitive received by the NWK layer
7942 shall contain a 16-bit logical address unique to that network which the child can use in future transmissions.
7943 The NWK layer shall then set the Relationship field in the corresponding neighbor table entry to indicate
7944 that the neighbor is its parent. By this time, the parent shall have added the new device to its neighbor table.
7945 Furthermore, the NWK layer will update the values of *nwkNetworkAddress*, *nwkUpdateId* and *mwkPANId*
7946 in the NIB.

7947 If the device is attempting to join a secure network and it is a router, it will need to wait until its parent has
7948 authenticated it before transmitting beacons. The device shall therefore wait for an
7949 NLME-START-ROUTER.request primitive to be issued from the next higher layer. Upon receipt of this
7950 primitive, the NLME shall issue an MLME-START.request primitive if it is a router. If the
7951 NLME-START-ROUTER.request primitive is issued on an end device, the NWK layer shall issue an
7952 NLME-START-ROUTER.confirm primitive with the status value set to INVALID_REQUEST.

7953 Once the device has successfully joined the network, if it is a router and the next higher layer has issued a
7954 NLME-START-ROUTER.request, the NWK layer shall issue the MLME-START.request primitive to its
7955 MAC sub-layer. The PANId, LogicalChannel, BeaconOrder and SuperframeOrder parameters shall be set
7956 equal to the corresponding values held in the neighbor table entry for its parent. The network depth is set to
7957 one more than the parent network depth unless the parent network depth has a value of 0x0f, *i.e.* the maximum
7958 value for the 4-bit device depth field in the beacon payload. In this case, the network depth shall also
7959 be set to 0x0f. The PANCoordinator and CoordRealignment parameters shall both be set to FALSE. Upon
7960 receipt of the MLME-START.confirm primitive, the NWK layer shall issue an
7961 NLME-START-ROUTER.confirm primitive with the same status value.

Figure 3.36 Procedure for Joining a Network Through Association



7964

3.6.1.4.1.2 Parent Procedure

7965
7966
7967
7968
7969

The procedure for a ZigBee coordinator or router to join a device to its network using the MAC sub-layer association procedure is initiated by the MLME-ASSOCIATE.indication primitive arriving from the MAC sub-layer. Only those devices that are either a ZigBee coordinator or a ZigBee router and that are permitting devices to join the network shall initiate this procedure. If this procedure is initiated on any other device, the NLME shall terminate the procedure.

7970
7971
7972
7973
7974
7975
7976
7977
7978
7979

When this procedure is initiated, the NLME of a potential parent shall first determine whether the device wishing to join already exists on its network. To do this, the NLME shall search its neighbor table in order to determine whether a matching 64-bit, extended address can be found. If an extended address match is found, the NLME shall check that the supplied DeviceCapabilities match the device type on record in the neighbor table. If the device type also matches the NLME, it shall then obtain the corresponding 16-bit network address and issue an association response to the MAC sub-layer. If a device type match is not found the NLME shall remove all records of the device in its neighbor table and restart processing of the MLME-ASSOCIATION.indication. If an extended address match is not found, the NLME shall, if possible, allocate a 16-bit network address for the new device. See section 3.6.1.6 and section 3.6.1.7 for an explanation of the address assignment mechanisms.

7980
7981
7982

If the potential parent does not have the capacity to accept more children, the NLME shall terminate the procedure and indicate this fact in the subsequent MLME-ASSOCIATE.response primitive to the MAC sub-layer. The Status parameter of this primitive shall indicate that the PAN is at capacity.

7983
7984
7985
7986
7987
7988
7989

If the request to join is granted, the NLME of the parent shall create a new entry for the child in its neighbor table using the supplied device information and indicate a successful association in the subsequent MLME-ASSOCIATE.response primitive to the MAC sub-layer. If the value of *nwkSecurityLevel* is 0x00, the relationship field of the new neighbor table entry shall be set to the value 0x01 indicating that the neighbor is a child; otherwise, it shall be set to 0x05 indicating an unauthenticated child. The status of the response transmission to the child is communicated back to the network layer via the MLME-COMM-STATUS.indication primitive.

7990
7991
7992
7993

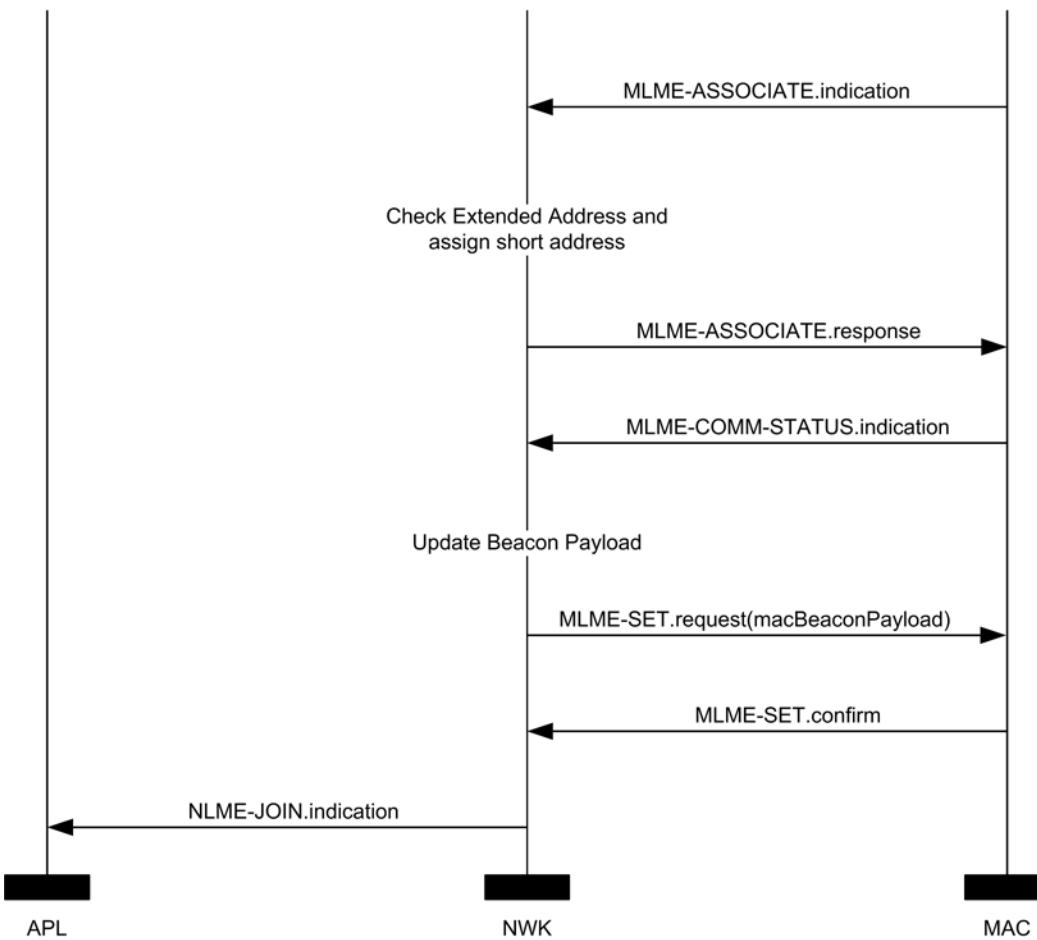
If the transmission was unsuccessful (*i.e.* the MLME-COMM-STATUS.indication primitive contained a Status parameter not equal to SUCCESS), the NLME shall terminate the procedure. If the transmission was successful, the NLME shall notify the next higher layer that a child has just joined the network by issuing the NLME-JOIN.indication primitive.

7994
7995

The procedure for successfully joining a device to the network is illustrated in the MSC shown in Figure 3.37.

7996

Figure 3.37 Procedure for Handling a Join Request



7997

7998

3.6.1.4.2 Joining or Rejoining a Network Using NWK Rejoin

7999
8000
8001
8002
8003
8004
8005

Devices that have lost all connection to the network, for example a ZED that can no longer communicate successfully with its parent, can rejoin the network using the NWK rejoin request and NWK rejoin response commands. The rejoining procedure is identical to the association procedure described in the previous section, except that the MAC association procedure is replaced by an exchange involving the rejoin request and rejoin response commands, and, because NWK commands make use of NWK security, no authentication step is performed. Using these commands instead of the MAC procedure allows a device to rejoin a network that does not currently allow new devices to join.

8006
8007

Devices that are joining a network for the first time may also use a variant of this procedure as described in the following sections.

8008

3.6.1.4.2.1 Child Procedure

8009
8010
8011
8012
8013
8014
8015
8016

The procedure for joining or rejoining a network using the NWK rejoin procedure shall be initiated by issuing the NLME-JOIN.request primitive, as shown in Figure 3.38, with the RejoinNetwork parameter set to 0x02 and the ExtendedPANId parameter set to the ExtendedPANId of the network to rejoin. The device type field of the CapabilityInformation parameter shall be set to 1 if the device is intended to join as a router and to 0 otherwise. If the value of the *nwkNetworkAddress* value in the NIB is within the valid range defined for that value, it shall use *nwkNetworkAddress* when issuing the Rejoin Request command. If the *nwkNetworkAddress* is NOT within the valid range, it shall randomly generate a short address within the valid range, excluding the value of 0x0000, and use that for the Rejoin Request command.

8017 The ScanChannels parameter shall be set to indicate which channels are to be scanned to locate this net-
8018 work and the ScanDuration parameter set to indicate the length of time to be spent scanning each channel.

8019 Upon receipt of this primitive, the NWK layer shall issue an MLME- SCAN.request primitive asking the
8020 MAC sub-layer to perform an active scan.

8021 Every beacon frame received during the scan having a non-zero length payload shall cause the
8022 MLME-BEACON-NOTIFY.indication primitive to be issued from the MAC sub-layer of the scanning de-
8023 vice to its NLME. The NLME of the scanning device shall check the ExtendedPANId contained within the
8024 beacon payload to see if it is of the correct value. If not, the beacon is ignored. Otherwise, the device shall
8025 copy the relevant information from each received beacon (see Figure 3.51 for the structure of the beacon
8026 payload) into its neighbor table (see Table 3.53 and Table 3.54 for the contents of a neighbor table entry).

8027 Once the MAC sub-layer signals the completion of the scan by issuing the MLME-SCAN.confirm primitive
8028 to the NLME, the NWK layer shall search its neighbor table for a suitable parent device. A suitable
8029 parent device shall advertise device capacity of the type requested in the JoinAsRouter parameter, shall
8030 have the most recent update id, where the determination of most recent update id must take into account
8031 that the update id will wrap back to zero, and shall have a link cost (see section 3.6.3.1) of 3, at most. If the
8032 neighbor table contains no devices that are suitable parents, the NLME shall respond with an
8033 NLME-JOIN.confirm with a Status parameter of NOT_PERMITTED. If the neighbor table has more than
8034 one device that could be a suitable parent, the device which is at a minimum depth from the ZigBee coor-
8035 dinator shall be chosen.

8036 Once a suitable parent is identified the device shall set its *nwkParentInformation* value in the NIB to 0, then
8037 the NLME shall construct a NWK rejoin request command frame. The destination address field of the
8038 NWK header shall have a value equal to the 16-bit network address of the parent candidate chosen from the
8039 neighbor table. The source address field of the NWK header shall be set to the value of the *nwkNetwork-*
8040 *Address* attribute of the NIB. Both the source IEEE address field and the destination IEEE address field
8041 shall be present in the NWK header. If the device is joining this network for the first time, and the value of
8042 the *nwkNetworkAddress* attribute of its NIB has a value of 0xffff indicating that it is not currently joined to
8043 a network, the device shall select a 16-bit network address for itself and set the *nwkNetworkAddress* attrib-
8044 ute to this value. The address should be randomly selected according to the procedures outlined in section
8045 3.6.1.7. In this case, and in any case where the *nwkAddrAlloc* attribute of the NIB has a value of 0x02 indi-
8046 cating stochastic addressing, the allocate address sub-field of the capability information field of the com-
8047 mand payload shall be set to 0 indicating a self-selected address.

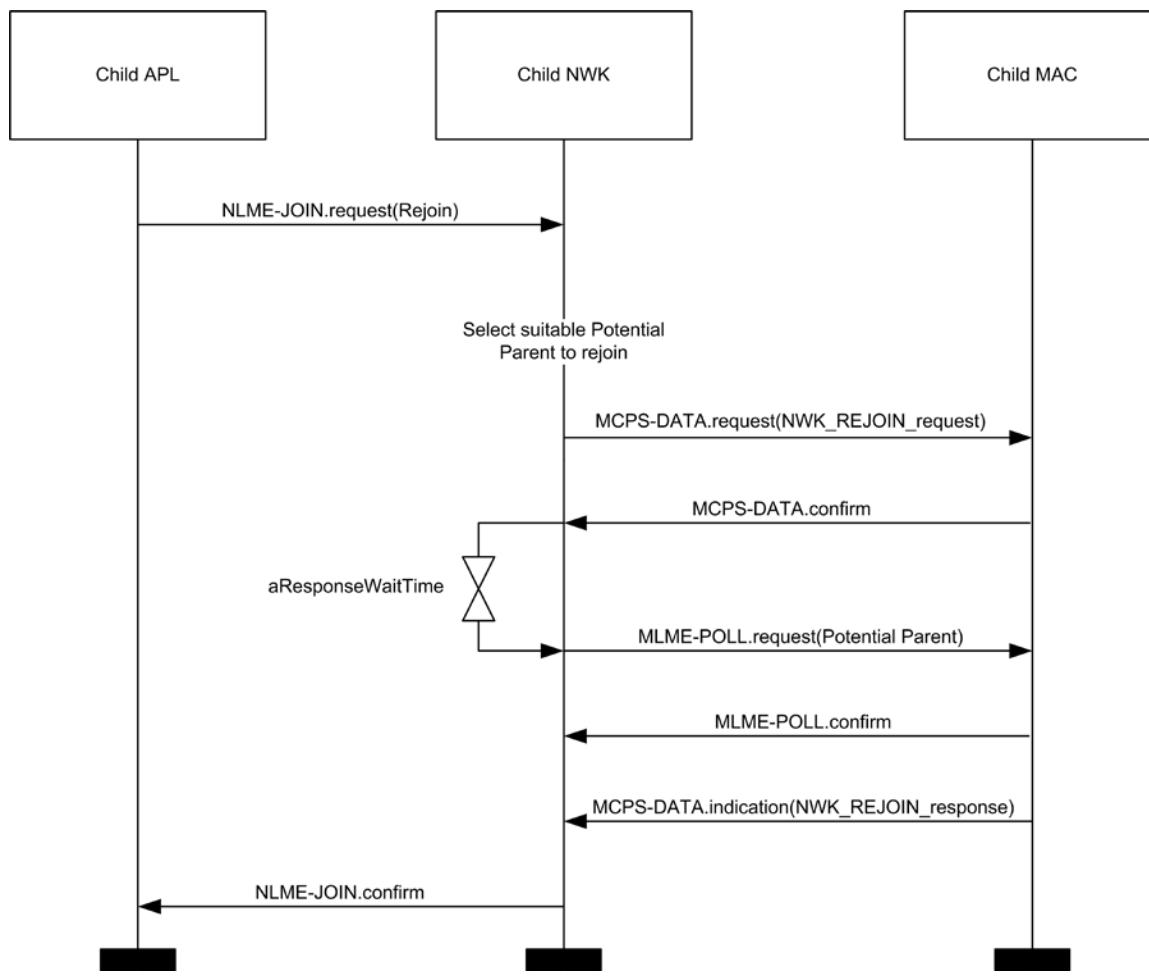
8048 After the successful transmission of the rejoin request command using the MAC data service, the network
8049 layer shall load a countdown timer with a value of *aResponseWaitTime* ([B1]). If this timer elapses before a
8050 rejoin response command frame is received, then the rejoin was unsuccessful. If the receiver on when idle
8051 field of the CapabilityInformation parameter is equal to 0, the device shall issue a MLME-POLL.request to
8052 the potential parent to retrieve the rejoin response command. If the receiver on when idle field is equal to 1,
8053 polling is not required.

8054 Note: Polling more than once before *aResponseWaitTime* ([B1]) elapses is permitted.

8055 On receipt of a rejoin response command frame, after the above procedure or at any other time, the device
8056 shall check the destination IEEE address field and the source IEEE address fields of the command frame
8057 NWK header. If the destination IEEE address field is not equal in value to the IEEE address of the receiv-
8058 ing device or if the source IEEE address field is not equal in value to the IEEE address of the most recent
8059 potential parent to which a rejoin request command frame was sent (or the current parent in the case of an
8060 unsolicited rejoin response), then the rejoin response command frame shall be discarded without further
8061 processing.

If the rejoin status field within the rejoin response command frame indicates a refusal to permit rejoining on the part of the neighboring device (that is, PAN at capacity or PAN access denied), then the device attempting to rejoin should set the potential parent bit to 0 in the corresponding neighbor table entry to indicate a failed join attempt. Setting the potential parent bit to 0 ensures that the NWK layer will not issue another request to rejoin to the same neighboring device. If the attempt to join was unsuccessful, the NLME shall attempt to find another suitable parent from the neighbor table. If no such device can be found, the NLME shall issue the NLME-JOIN.confirm primitive with the Status parameter set to NOT_PERMITTED. If the attempt to join is unsuccessful and there is a second neighboring device that could be a suitable parent, the NWK layer shall initiate the NWK rejoin procedure with the second device. The NWK layer shall repeat this procedure until it either rejoins the PAN successfully or exhausts its options to rejoin the PAN. If the device cannot successfully rejoin the PAN specified by the next higher layer, the NLME shall terminate the procedure by issuing the NLME-JOIN.confirm primitive with the Status parameter set to NOT_PERMITTED. In this case, the device shall not receive a valid logical address and shall not be permitted to transmit on the network. If the attempt to rejoin was successful, the NWK rejoin response command received by the NWK layer shall contain a 16-bit logical address unique to that network, which the child can use in future transmissions. Note that this address may be identical to the current 16-bit network address of the device stored in the *nwkNetworkAddress* attribute of the NIB. The NWK layer shall then set the relationship field in the corresponding neighbor table entry to indicate that the neighbor is its parent. By this time, the parent shall have added the new device to its neighbor table. Furthermore, the NWK layer shall update the values of *nwkNetworkAddress*, *nwkUpdateId*, and *nwkPANId* in the NIB if necessary.

Figure 3.38 Child Rejoin Procedure



8085

3.6.1.4.2.2 Parent Procedure

8086

The procedure for a ZigBee coordinator or router to rejoin a device to its network using the NWK rejoin procedure is initiated by the arrival of a NWK layer rejoin command frame via the MAC data service. Only those devices that are either ZigBee coordinators or ZigBee routers shall initiate this procedure. If this procedure is initiated on any other device, the NLME shall terminate the procedure. When this procedure is initiated, the NLME of a potential parent shall first determine whether it already has knowledge of the requesting device. To do this, the NLME shall search its neighbor table in order to determine whether a matching 64-bit, extended address can be found. If an extended address match is found, the NLME shall check that the supplied DeviceCapabilities match the device type on record in the neighbor table. If the device type matches, the NLME shall consider the join attempt successful and use the 16-bit network address found in its neighbor table as the network address of the joining device. If a device type match is not found, the NLME shall remove all records of the device in its neighbor table and restart processing of the NWK layer rejoin command.

8098

If the potential parent does not have the capacity to accept the joining device, the NLME shall terminate the procedure and indicate this fact in the subsequent rejoin response command. The Status parameter of this command shall indicate that the PAN is at capacity.

8101

If the request to rejoin is granted, the NLME of the parent shall create a new entry for the child in its neighbor table, or modify the existing entry if one such already exists, using the supplied device information, and indicate a successful rejoin by replying to the requesting device with a NWK rejoin response command. If the *nwkAddrAlloc* attribute of the NIB has a value of 0x00, indicating tree addressing, the NLME shall allocate new a 16-bit network address for the joining device. See section 3.6.1.6 and section 3.6.1.7 for an explanation of the address assignment mechanisms.

8107

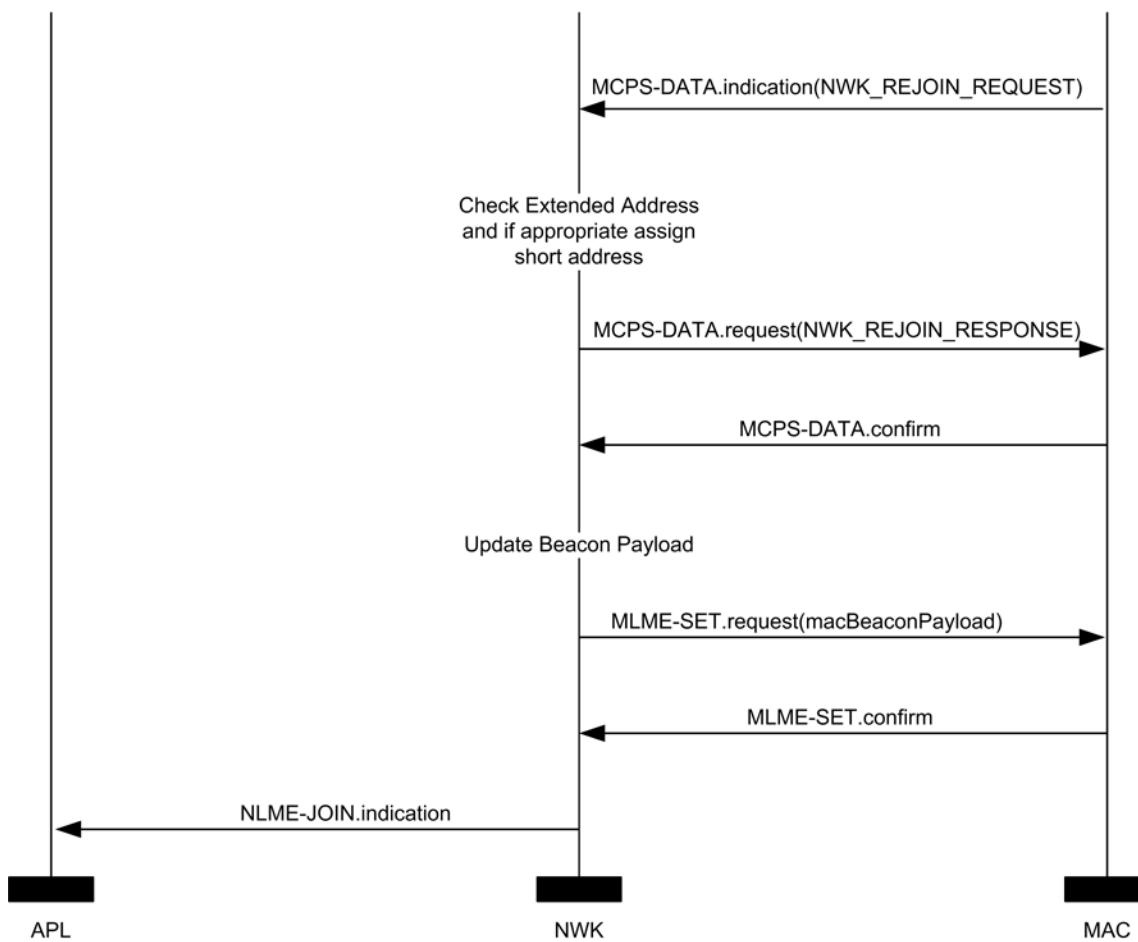
If the *nwkAddrAlloc* attribute of the NIB does not have a value of 0x00, the allocate address sub-field of the capabilities information field of the rejoin request command frame payload may have a value of 0 indicating a self-assigned or pre-existing network address. In this case, as is the case with all NWK command frames, the 16-bit network address in the source address field of the NWK header, in combination with the 64-bit IEEE address from the source IEEE address field of the network header should be checked for address conflicts as described in section 3.6.1.9. If an address conflict is discovered, a new, and non-conflicting, address shall be chosen for the joining device and shall be placed in the network address field of command frame payload of the outgoing rejoin response command frame. Otherwise, the contents of the source address field of the incoming rejoin request command frame shall be placed in the network address field of the command frame payload of the outgoing rejoin response command frame.

8117

The NLME shall then notify the next higher layer that a child has just rejoined the network by issuing the NLME-JOIN.indication primitive. The procedure for successfully rejoining a device to the network is illustrated in the MSC shown in Figure 3.39.

8120

Figure 3.39 Parent Rejoin Procedure



8121

APL

8122

8123

3.6.1.4.3 Joining a Network Directly

8124 This section specifies how a device can be directly added to a network by a previously designated parent
8125 device (ZigBee coordinator or router). In this case, the parent device is preconfigured with the 64-bit ad-
8126 dress of the child device. The following text describes how this prior address knowledge may be used to
8127 establish the parent-child relationship.

8128 The procedure for a ZigBee coordinator or router to directly join a device to its network is initiated by is-
8129 suing the NLME-DIRECT-JOIN.request primitive with the DeviceAddress parameter set to the address of
8130 the device to be joined to the network. Only those devices that are either a ZigBee coordinator or a ZigBee
8131 router may initiate this procedure. If this procedure is initiated on any other device, the NLME may termi-
8132 nate the procedure and notify the next higher layer of the illegal request by issuing the
8133 NLME-DIRECT-JOIN.confirm primitive with the Status parameter set to INVALID_REQUEST.

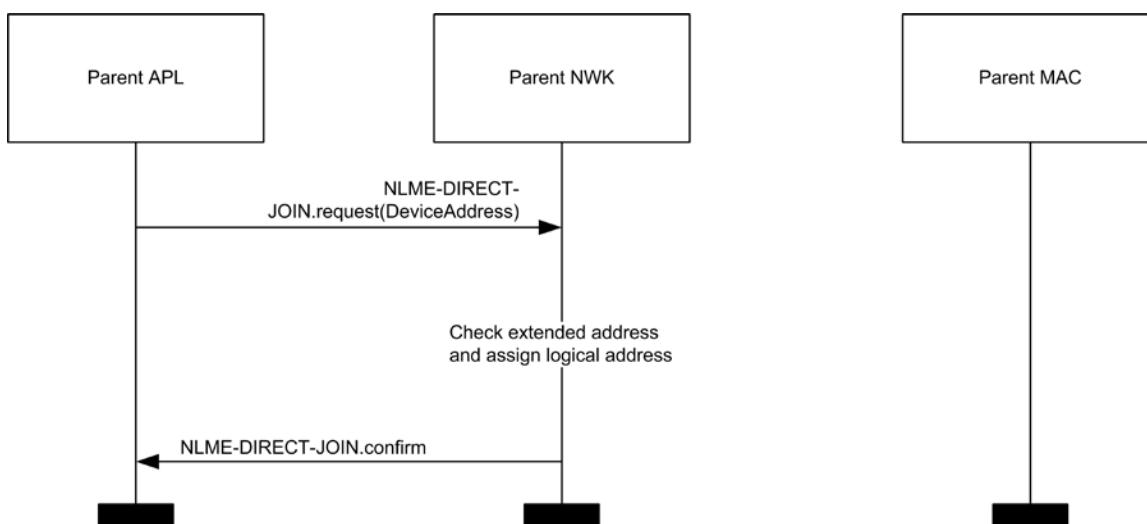
8134 When this procedure is initiated, the NLME of the parent shall first determine whether the specified device
8135 already exists on its network. To do this, the NLME shall search its neighbor table in order to determine
8136 whether a matching 64-bit, extended address can be found. If a match is found, the NLME shall terminate
8137 the procedure and notify the next higher layer that the device is already present in the device list by issuing
8138 the NLME-DIRECT-JOIN.confirm primitive with the Status parameter set to ALREADY_PRESENT.

8139 If a match is not found, the NLME shall, if possible, allocate a 16-bit network address for the new device as
8140 well as a new neighbor table entry. See section 3.6.1.6 and section 3.6.1.7 for an explanation of the address
8141 assignment mechanisms. If the parent device has no more room in its neighbor table, the NLME shall ter-
8142 minate the procedure and notify the next higher layer of the unavailable capacity by issuing the
8143 NLME-DIRECT-JOIN.confirm primitive with the Status parameter set to NEIGHBOR_TABLE_FULL. If
8144 capacity is available, the NLME shall inform the next higher layer that the device has joined the network by
8145 issuing the NLME-DIRECT-JOIN.confirm primitive with the Status parameter set to SUCCESS.

8146 Once the parent has added the child to its network, it is still necessary for the child to make contact with the
8147 parent to complete the establishment of the parent-child relationship. The child shall fulfill this require-
8148 ment by initiating the orphaning procedure, which is described in section 3.6.1.4.3.1.

8149 A parent that supports direct joining shall follow the procedure illustrated in Figure 3.40 to successfully
8150 join a device to the network directly. This procedure does not require any over-the-air transmissions.

8151 **Figure 3.40 Joining a Device to a Network Directly**



8152

8153

8154 3.6.1.4.3.1 Joining or Re-joining a Network Through Orphaning

8155 This section specifies how the orphaning procedure can be initiated by a device that has been directly
8156 joined to a network (joining through orphaning) or by a device that was previously joined to a network but
8157 has lost contact with its parent (re-joining through orphaning).

8158 A device that has been added to a network directly shall initiate the orphan procedure in order to complete
8159 the establishment of its relationship with its parent. The application on the device will determine whether to
8160 initiate this procedure and, if so, will notify the network layer upon power up.

8161 A device that was previously joined to a network has the option of initiating the orphan procedure if its
8162 NLME repeatedly receives communication failure notifications from its MAC sub-layer.

8163 3.6.1.4.3.2 Child Procedure

8164 The optional joining through orphaning procedure is initiated by a device using the NLME-JOIN.request
8165 primitive with the RejoinNetwork parameter set to 0x01.

8166 When this procedure is initiated, the NLME shall first request that the MAC sub-layer perform an orphan
8167 scan over the set of channels given by the ScanChannels parameter. An orphan scan is initiated by
8168 issuing the MLME-SCAN.request primitive to the MAC sub-layer, and the result is communicated back to
8169 the NLME via the MLME-SCAN.confirm primitive.

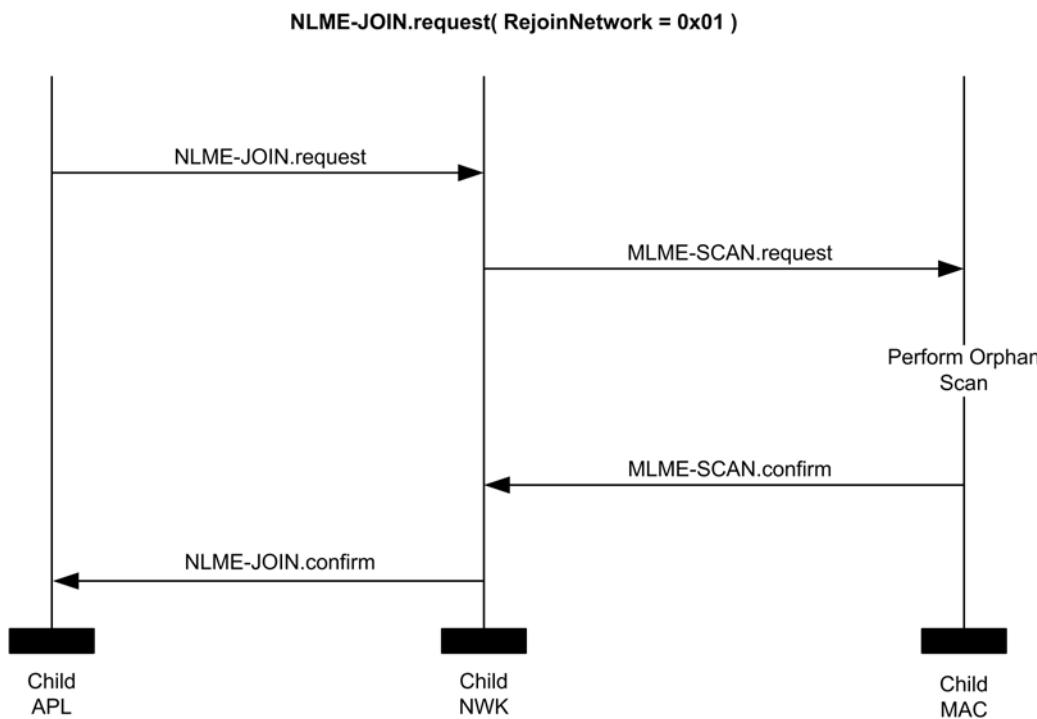
8170 If the child has found its parent, the orphan scan was successful and the NLME shall inform the next higher
8171 layer of the success of its request to join or re-join the network by issuing the NLME-JOIN.confirm primitive
8172 with the Status parameter set to SUCCESS.

8173 Note that if the child device is joining for the first time or if the child device has previously been joined to
8174 the network, but has failed to retain tree depth information as prescribed in section 3.6.1.8, it may not be
8175 able to operate correctly on the network without taking measures, outside the scope of this specification, for
8176 the recovery of this information.

8177 If the orphan scan was unsuccessful (the parent has not been found), the NLME shall terminate the proce-
8178 dure and notify the next higher layer that no networks were found. This is achieved by issuing the
8179 NLME-JOIN.confirm primitive with the Status parameter set to NO_NETWORKS.

8180 The procedure for a child to successfully join or re-join a network through orphaning is illustrated in the
8181 MSC shown in Figure 3.41.

8182 **Figure 3.41 Child Procedure for Joining or Re-Joining a Network through Orphaning**



3.6.1.4.3.3 Parent Procedure

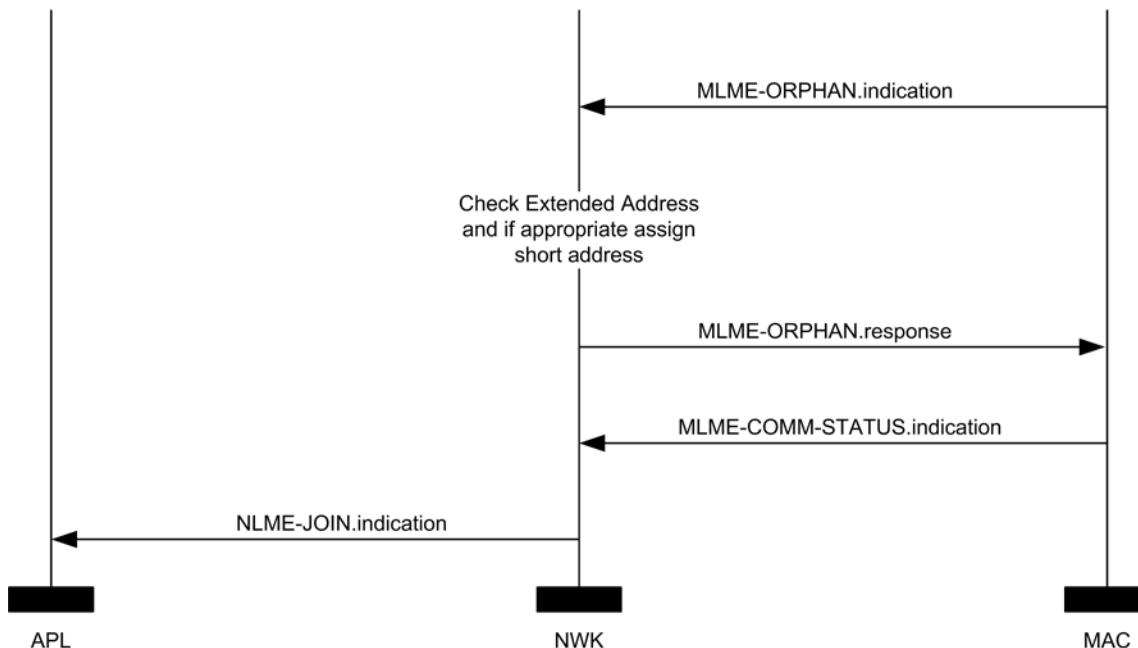
8186 A device is notified of the presence of an orphaned device when it receives the
8187 MLME-ORPHAN.indication primitive from the MAC sub-layer. Only devices that are either ZigBee coor-
8188 dinators or ZigBee routers (that is, devices with parental capabilities) shall initiate this procedure. If this
8189 procedure is initiated by any other device, the NLME shall terminate the procedure.

8190 When this procedure is initiated, the NLME shall first determine whether the orphaned device is its child.
8191 This is accomplished by comparing the extended address of the orphaned device with the addresses of its
8192 children, as recorded in its neighbor table. If a match is found (the orphaned device is its child), the NLME
8193 shall obtain the corresponding 16-bit network address and include it in its subsequent orphan response to
8194 the MAC sub-layer. The orphan response to the MAC sub-layer is initiated by issuing the
8195 MLME-ORPHAN.response primitive, and the status of the transmission is communicated back to the
8196 NLME via the MLME-COMM-STATUS.indication primitive.

8197 If an address match is not found (the orphaned device is not its child), the procedure shall be terminated
8198 without indication to the higher layer.

8199 The procedure for a parent to join or re-join its orphaned child to the network is illustrated in the MSC
8200 shown in Figure 3.42.

8201 **Figure 3.42 Parent Procedure for Joining or Re-Joining a Device to Its Network through Orphaning**



8202
8203

3.6.1.5 Neighbor Tables

8204 The neighbor table of a device shall contain information on every device within transmission range, up to
8205 some implementation-dependent limit.

8206 The neighbor table is useful in two contexts. First of all, it is used during network discovery or rejoining to
8207 store information about routers within RF reception range that may be candidate parents. Second, after the
8208 device has joined a network, it is used to store relationship and link-state information about neighboring
8209 devices in that network. A table entry shall be updated every time a device receives any frame from the
8210 corresponding neighbor.

8211 The outgoing cost field contains the cost of the link as measured by the neighbor. The value is obtained
8212 from the most recent link status command frame received from the neighbor. A value of 0 indicates that no
8213 link status command listing this device has been received.

8214 The age field indicates the number of *nwkLinkStatusPeriod* intervals that have passed since the last link
8215 status command frame was received, up to a maximum value of *nwkRouterAgeLimit*.

8216 Mandatory and optional data that are used in normal network operation are listed in Table 3.53.

Table 3.53 Neighbor Table Entry Format

Field Name	Field Type	Valid Range	Description
Extended address	Integer	An extended 64-bit, IEEE address	64-bit IEEE address that is unique to every device.
Network address	Network address	0x0000 – 0xffff7	The 16-bit network address of the neighboring device. This field shall be present in every neighbor table entry.
Device type	Integer	0x00 – 0x02	The type of neighbor device: 0x00 = ZigBee coordinator 0x01 = ZigBee router 0x02 = ZigBee end device This field shall be present in every neighbor table entry.
RxOnWhenIdle	Boolean	TRUE or FALSE	Indicates if neighbor's receiver is enabled during idle periods: TRUE = Receiver is on FALSE = Receiver is off This field should be present for entries that record the parent or children of a ZigBee router or ZigBee coordinator.
End Device Configuration	Bitmask	0x0000 – 0xFFFF	The end device's configuration. See Error! Reference source not found. section 3.4.11.3.2. The default value shall be 0.
Timeout Counter	Integer	0x00000000 – 0x00F00000	This field indicates the current time remaining, in seconds, for the end device.

Field Name	Field Type	Valid Range	Description
Device Timeout	Integer	0x00000000 – 0x0001FA40	<p>This field indicates the timeout, in seconds, for the end device child.</p> <p>The default value for end device entries is calculated by using the <i>nwkEndDeviceTimeoutDefault</i> value and indexing into Table 3.44, then converting the value to seconds. End Devices may negotiate a longer or shorter time using the NWK Command End Device Timeout Request.</p>
Relationship	Integer	0x00 – 0x05	<p>The relationship between the neighbor and the current device:</p> <ul style="list-style-type: none"> 0x00=neighbor is the parent 0x01=neighbor is a child 0x02=neighbor is a sibling 0x03=none of the above 0x04=previous child 0x05=unauthenticated child <p>This field shall be present in every neighbor table entry.</p>
Transmit Failure	Integer	0x00 – 0xff	<p>A value indicating if previous transmissions to the device were successful or not. Higher values indicate more failures.</p> <p>This field shall be present in every neighbor table entry.</p>
LQI	Integer	0x00 – 0xff	<p>The estimated link quality for RF transmissions from this device. See section 3.6.3.1 for a discussion of how this is calculated.</p> <p>This field shall be present in every neighbor table entry.</p>
Outgoing Cost	Integer	0x00 - 0xff	<p>The cost of an outgoing link as measured by the neighbor. A value of 0 indicates no outgoing cost is available.</p> <p>This field is mandatory if <i>nwkSymLink</i> = TRUE.</p>

Field Name	Field Type	Valid Range	Description
Age	Integer	0x00 - 0xff	The number of nwkLinkStatusPeriod intervals since a link status command was received. This field is mandatory if nwkSymLink = TRUE.
Incoming beacon timestamp	Integer	0x000000-0xfffffff	The time, in symbols, at which the last beacon frame was received from the neighbor. This value is equal to the timestamp taken when the beacon frame was received, as described in IEEE 802.15.4-2003 [B1]. This field is optional.
Beacon transmission time offset	Integer	0x000000-0xfffffff	The transmission time difference, in symbols, between the neighbor's beacon and its parent's beacon. This difference may be subtracted from the corresponding incoming beacon timestamp to calculate the beacon transmission time of the neighbor's parent. This field is optional.
Keepalive Received	Boolean	TRUE or FALSE	This value indicates at least one keepalive has been received from the end device since the router has rebooted.

8219

8220 Information that may be used during network discovery and rejoining, as described above, is shown in Ta-
 8221 ble 3.54. All of the fields shown are optional and should not be retained after the NLME has chosen a net-
 8222 work to join. Neighbor table entries corresponding to devices that are not members of the chosen network
 8223 should similarly be discarded.

Table 3.54 Additional Neighbor Table Fields

Field Name	Field Type	Valid Range	Description
Extended PAN ID	Integer	0x0000000000000001 - 0xfffffffffffffe	The 64-bit unique identifier of the network to which the device belongs.
Logical channel	Integer	Selected from the available logical channels supported by the PHY.	The logical channel on which the network is operating.
Depth	Integer	0x00 – 0x0f	The tree depth of the neighbor device.
Beacon order	Integer	0x00 – 0x0f	The IEEE 802.15.4 beacon order for the device.
Permit joining	Boolean	TRUE or FALSE	An indication of whether the device is accepting joining requests. TRUE = device is accepting join requests. FALSE = device is not accepting join requests.
Potential parent	Integer	0x00 – 0x01	An indication of whether the device has been ruled out as a potential parent. 0x00 indicates that the device is not a potential parent. 0x01 indicates that the device is a potential parent.

3.6.1.6 Distributed Address Assignment Mechanism

The default value of the NIB attribute *nwkAddrAlloc* is 0x00, where network addresses are assigned using a distributed addressing scheme that is designed to provide every potential parent with a finite sub-block of network addresses. These addresses are unique within a particular network and are given by a parent to its children. The ZigBee coordinator determines the maximum number of children any device, within its network, is allowed. Of these children, a maximum of *nwkMaxRouters* can be router-capable devices. The remaining devices shall be reserved for end devices. Every device has an associated depth that indicates the minimum number of hops a transmitted frame must travel, using only parent-child links, to reach the ZigBee coordinator. The ZigBee coordinator itself has a depth of 0, while its children have a depth of 1. Multi-hop networks have a maximum depth that is greater than 1. The ZigBee coordinator also determines the maximum depth of the network.

8236 Given values for the maximum number of children a parent may have, $nwkMaxChildren (Cm)$, the maxi-
8237 mum depth in the network, $nwkMaxDepth (Lm)$, and the maximum number of routers a parent may have as
8238 children, $nwkMaxRouters (Rm)$, we may compute the function, $Cskip(d)$, essentially the size of the address
8239 sub-block being distributed by each parent at that depth to its router-capable child devices for a given net-
8240 work depth, d , as follows:

$$8241 Cskip(d) = \begin{cases} 1 + Cm \cdot (Lm - d - 1), & \text{if } Rm = 1 \\ 8242 \frac{1 + Cm - Rm - Cm * Rm^{Lm-d-1}}{1 - Rm}, & \text{otherwise} \end{cases}$$

8243 If a device has a $Cskip(d)$ value of 0, then it shall not be capable of accepting children and shall be treated
8244 as a ZigBee end device for purposes of this discussion. The NLME of the device shall set the End device
8245 Capacity and Router Capacity sub fields of the MAC sub-layer beacon payload to 0.

8246 A parent device that has a $Cskip(d)$ value greater than 0 shall accept child devices and shall assign address-
8247 es to them differently depending on whether or not the child device is router-capable.

8248 Network addresses shall be assigned to router-capable child devices using the value of $Cskip(d)$ as an off-
8249 set. A parent assigns an address that is 1 greater than its own to its first router-capable child device. Subse-
8250 quently assigned addresses to router-capable child devices are separated from each other by $Cskip(d)$. A
8251 maximum of $nwkMaxRouters$ of such addresses shall be assigned.

8252 Network addresses shall be assigned to end devices in a sequential manner with the n^{th} address, , given by
8253 the following equation:

$$8254 A_n = A_{\text{parent}} + Cskip(d) * Rm + n$$

8255 Where $d(1 < n < (Cm - Rm))$ and A_{parent} represents the address of the parent.

8256 The $Cskip(d)$ values for an example network having $nwkMaxChildren=6$, $nwkMaxRouters=4$ and
8257 $nwkMaxDepth=3$ are calculated and listed in Table 3.55. Figure 3.43 illustrates the example network.

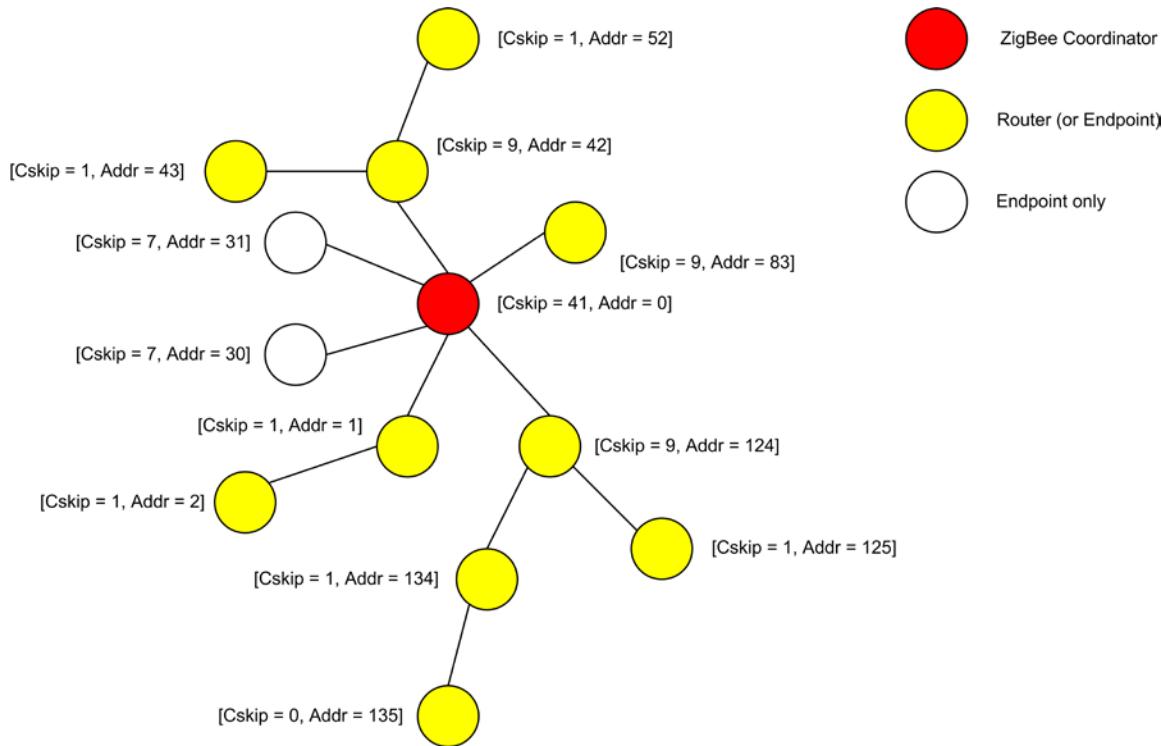
8258 **Table 3.55 Example Addressing Offset Values for Each Given Depth within the Network**

Depth in the Network, d	Offset Value, $Cskip(d)$
d of 0	Change $Cskip(d)$ to 31
d of 1	Change $Cskip(d)$ to 9
d of 2	Leave $Cskip(d)$ as 1
d of 3	Leave $Cskip(d)$ as 0

8260

8261

Figure 3.43 Address Assignment in an Example Network



Note: Can't use same Rm, Cm and Lm as spec. as this does not yield 'endpoint only' addresses.
Changed to Cm = 6 (from 4), Lm = 3, Rm = 4. This allows two extra 'endpoint only' devices to be shown at depth 1.

8262

Because an address sub-block cannot be shared between devices, it is possible that one parent exhausts its list of addresses while a second parent has addresses that go unused. A parent having no available addresses shall not permit a new device to join the network by setting the End Device Capacity and Router Capacity sub fields of the MAC sub-layer beacon payload to 0.

8267
8268
8269

In this situation, the new device shall find another parent. If no other parent is available within transmission range of the new device, the device shall be unable to join the network unless it is physically moved or there is some other change.

8270

3.6.1.7 Stochastic Address Assignment Mechanism

8271
8272
8273
8274
8275
8276
8277
8278
8279

When the NIB attribute *nwkAddrAlloc* has value 0x02, addresses shall be chosen at random. The value of *nwkMaxRouter* is not relevant in this case. The random address assigned shall conform to the NIST testing regimen described in reference [B12]. When a device joins the network using MAC association, its parent shall choose a random address that does not already appear in any entry in the parent's NIB. Under stochastic addressing, once a device has been assigned an address, it has no reason to relinquish that address and should retain it unless it receives an indication that its address is in conflict with that of another device on the network. Furthermore, devices may self-assign random addresses under stochastic addressing and retain them, as in the case of joining a network using the rejoin command frame (see section 3.6.1.4.2). The ZigBee coordinator, which has no parent, shall always have the address 0x0000.

3.6.1.8 Installation and Addressing

It should be clear that *nwkMaxDepth* roughly determines the number of hops in network terms from the root of the tree to the farthest end device. In principle, *nwkMaxDepth* also determines the overall network diameter. In particular, for an ideal network layout in which the ZigBee coordinator is located in the center of the network, as illustrated in Figure 3.43, the network diameter should be $2 * nwkMaxDepth$. In practice, application-driven placement decisions and order of deployment may lead to a smaller diameter. In this case, *nwkMaxDepth* provides a lower bound on the network diameter while the $2 * nwkMaxDepth$ provides the upper bound.

Finally, due to the fact that the tree is not dynamically balanced, when *nwkAddrAlloc* has a value of 0x00, the possibility exists that certain installation scenarios, such as long lines of devices, may exhaust the address capacity of the network long before the real capacity is reached.

Under stochastic address assignment, *nwkMaxDepth* is related to the number of hops across the network. This is not a controlled value in networks using stochastic address assignment.

3.6.1.9 Address Conflicts

An address conflict occurs when two devices in the same network have identical values for *nwkNetworkAddress*. Preventing all such conflicts, for example by using tree address assignment and prohibiting the reuse of assigned addresses, is not always practical. This section describes how address conflicts that do occur can be detected and corrected. Address conflict detection shall be enabled if the NIB attribute *nwkUniqueAddr* is FALSE.

Note that the network addresses used in routing messages are verified during the route discovery process. The *device_annc* now is also used to verify addresses. The verification applies only to devices, links, and information present at the time of the discovery or *device_annc*. Verification can be achieved at other times, such as before sending a unicast directly to a neighbor, by sending a network status command with a status code value of 0x0e, indicating address verification.

If a device receives a broadcast data frame and discovers an address conflict as a result of the receipt, as discussed below in section 3.6.1.9.2, it should not retransmit the frame as usual but shall discard it before taking the resolution actions described below in section 3.6.1.9.3.

3.6.1.9.1 Obtaining Address Information

The NWK layer obtains address information from incoming messages, including both NWK commands and data messages. Address information from data messages is passed to the NWK layer by being added to the network address map table in the NIB.

The ability to detect address conflicts is enhanced by adding one or both of the Destination IEEE Address and Source IEEE Address fields to a message's NWK frame. When *nwkUniqueAddr* is FALSE, all NWK command messages shall contain the source IEEE address and also the destination IEEE address if it is known by the source device.

When *nwkUniqueAddr* is FALSE, route request commands shall include the sender's IEEE address in the Sender IEEE address field. This ensures that devices are aware of their neighbors' IEEE addresses.

3.6.1.9.2 Detecting Address Conflicts

After joining a network or changing address due to a conflict, a device shall send either a *device_annc* or initiate a route discovery prior to sending messages.

Upon receipt of a frame containing a 64-bit IEEE address in the NWK header, the contents of the *nwkAddressMap* attribute of the NIB and neighbor table should be checked for consistency.

If the destination address field of the NWK Header of the incoming frame is equal to the *nwkNetworkAddress* attribute of the NIB then the NWK layer shall check the destination IEEE address field, if present and even if it is the 0xffffffffffff address, against the value of *aExtendedAddress*. If the IEEE addresses are not identical then a local address conflict has been detected on *nwkNetworkAddress*.

8326 If a neighbor table or address map entry is located in which the 64-bit address is the null IEEE address
8327 (0x00....00), the 64-bit address in the table can be updated. However, if the 64-bit address is not the null
8328 IEEE address and does not correspond to the received 64-bit address, the device has detected a conflict
8329 elsewhere in the network.

8330 When a broadcast frame is received that creates a new BTR, if the Source Address field in the NWK Header
8331 is equal to the *nwkNetworkAddress* attribute of the NIB then a local address conflict has been detected on
8332 *nwkNetworkAddress*.

8333 Address conflicts are resolved as described in section 3.6.1.9.3.

3.6.1.9.3 Resolving Address Conflicts

8335 If a ZigBee coordinator or Router determines that there are multiple users of an address that is not its own,
8336 it shall inform the network by broadcasting a network status command with a status code of 0x0d indicating
8337 address conflict, and with the offending address in the destination address field. The network status com-
8338 mand shall be broadcast to 0xFFFFD, i.e. all devices with *macRxOnWhenIdle* = TRUE. The device shall de-
8339 lay initiation of this broadcast by a random jitter amount bounded by *nwkcMaxBroadcastJitter*. If during
8340 this delay a network status is received with the identical payload, the device shall cancel its own broadcast.

8341 If the device has learned of the conflict other than receiving a network status command with a status of
8342 0x0d, then it shall inform the network by broadcasting a network status command with a status code of
8343 0x0d indicating address conflict, and with its previous address in the destination address field. The network
8344 status command shall be broadcast to 0xFFFFD, i.e. all devices with *macRxOnWhenIdle*= TRUE. The de-
8345 vice shall delay initiation of this broadcast by a random jitter amount bounded by *nwkcMaxBroadcastJitter*.
8346 If during this delay a network status is received with the identical payload, the device shall cancel its own
8347 broadcast. Regardless of how it learned of the conflict, it shall implement the procedure on Detecting Ad-
8348 dress Conflicts detailed in section 3.6.1.9.2.

8349 If the conflict is detected on a ZigBee end device or *nwkAddrAlloc* is not equal to stochastic address as-
8350 signment then the device shall perform a rejoin to obtain a new address. Otherwise, the device that requires
8351 a new address shall pick a new address randomly, avoiding all addresses that appear in NIB entries.

8352 If a parent device detects or is informed of a conflict with the address of an end device child, the parent
8353 shall pick a new address for the end device child and shall send an unsolicited rejoin response command
8354 frame to inform the end device child of the new address. To notify the next higher layer of an address
8355 change the end device shall issue an NLME-NWK-STATUS.indication with status 'Network Address Up-
8356 date' and the new network address as the value of the ShortAddr parameter.

3.6.1.10 Leaving a Network

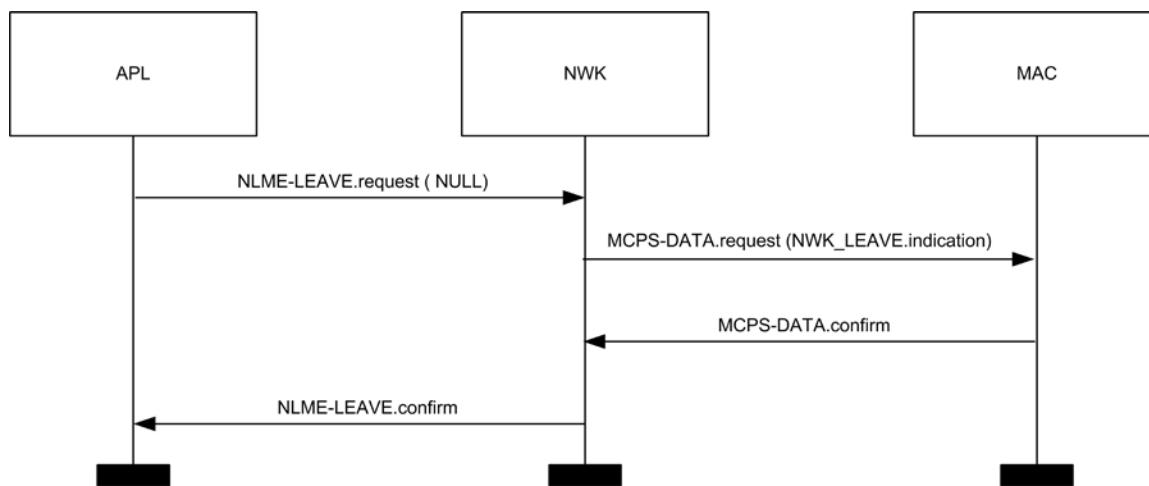
8357 This section specifies methods for a device to remove itself from the network and for the parent of a device
8358 to request its removal. In both cases, the children of the removed device, if any, may also be removed.

3.6.1.10.1 Method for a Device to Initiate Its Own Removal from the Network

8362 This section describes how a device can initiate its own removal from the network in response to the receipt
8363 of an NLME-LEAVE.request primitive from the next higher layer as shown in Figure 3.44.

8364

Figure 3.44 Initiation of the Leave Procedure



8365

8366

8367 When the NWK layer of a ZigBee router or ZigBee coordinator, receives the NLME-LEAVE.request primitive with the DeviceAddress parameter equal to NULL or equal to the local device's IEEE address (indicating that the device is to remove itself) the device shall send a leave command frame using the MCPS-DATA.request primitive with the DstAddr parameter set to 0xffff indicating a MAC broadcast. The request sub-field of the command options field of the leave command frame shall be set to 0. The value of the remove children sub-field of the command options field of the leave command shall reflect the value of the RemoveChildren parameter of the NLME-LEAVE.request primitive, and the value of the Rejoin sub-field of the leave command shall reflect the value of the Rejoin parameter of the NLME-LEAVE.request primitive. After transmission of the leave command frame, it shall issue a NLME-LEAVE.confirm primitive to the higher layer with the DeviceAddress parameter equal to NULL. The Status parameter shall be SUCCESS if the leave command frame was transmitted successfully. Otherwise, the Status parameter of the NLME-LEAVE.confirm shall have the same value as the Status parameter returned by the MCPS-DATA.confirm primitive. Regardless of the Status parameter to the NLME-LEAVE.confirm, the device shall leave the network employing the procedure in 3.6.1.10.4.

8368 If the device receiving the NLME-LEAVE.request primitive is a ZigBee end device, then the device shall
8369 send a leave command frame using the MCPS-DATA.request primitive with the DstAddr parameter set to
8370 the 16-bit network address of its parent device, indicating a MAC unicast. The request and remove children
8371 sub-fields of the command options field of the leave command frame shall be set to 0, and the rejoin flag in
8372 the command options shall be copied from the rejoin parameter of the NLME-LEAVE.request primitive.
8373 After transmission of the leave command frame, it shall set the *nwkExtendedPANId* attribute of the NIB to
8374 0x0000000000000000 and issue a NLME-LEAVE.confirm primitive to the higher layer with the De-
8375 viceAddress parameter equal to NULL. The Status parameter shall be SUCCESS if the leave command
8376 frame was transmitted successfully. Otherwise, the Status parameter of the NLME-LEAVE.confirm shall
8377 have the same value as the Status parameter returned by the MCPS-DATA.confirm primitive. Regardless
8378 of the Status parameter to the NLME-LEAVE.confirm, the device shall leave the network employing the
8379 procedure in 3.6.1.10.4.

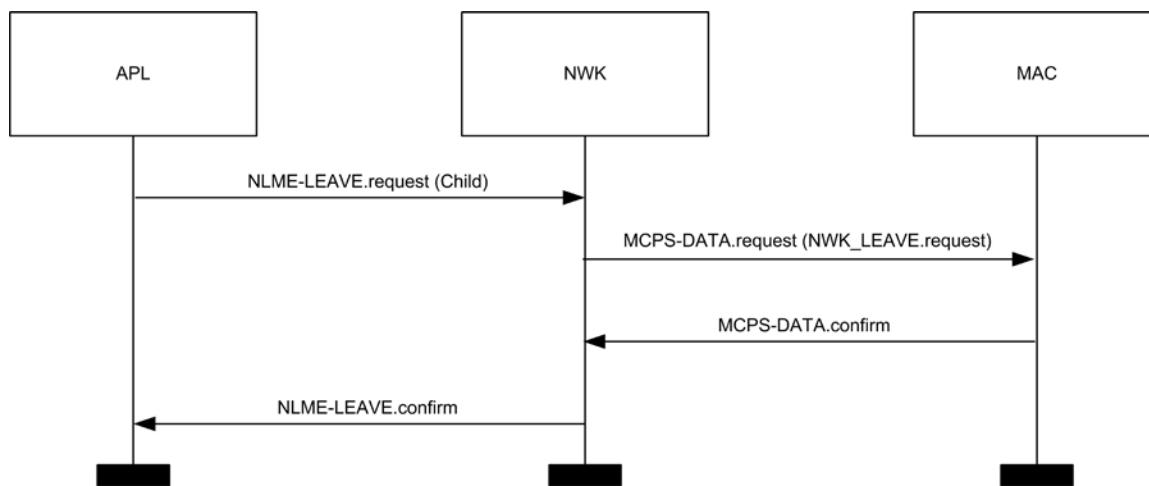
8393

3.6.1.10.2 Method for a Device to Remove Its Child from the Network

8394 This section describes how a device can initiate the removal from the network of one of its child devices in
8395 response to the receipt of an NLME-LEAVE.request primitive from the next higher layer as shown in Fig-
8396 ure 3.45.

8398

Figure 3.45 Procedure for a Device to Remove Its Child



8399

8400

8401 When the NWK layer of a ZigBee coordinator or ZigBee router, receives the NLME-LEAVE.request primitive
8402 with the DeviceAddress parameter equal to the 64-bit IEEE address of a child device, if the relation-
8403 ship field of the neighbor table entry corresponding to that child device does not have a value of 0x05 indi-
8404 cating that the child has not yet authenticated, the device shall send a network leave command frame using
8405 the MCPS-DATA.request primitive with the DstAddr parameter set to the 16-bit network address of that
8406 child device. The request sub-field of the command options field of the leave command frame shall have a
8407 value of 1, indicating a request to leave the network. The value of the remove children sub-field of the
8408 command options field of the leave command shall reflect the value of the RemoveChildren parameter of
8409 the NLME-LEAVE.request primitive, and the value of the Rejoin sub-field of the leave command shall re-
8410 flect the value of the Rejoin parameter of the NLME-LEAVE.request primitive.

8411 If the relationship field of the neighbor table entry corresponding to the device being removed has a value
8412 of 0x05, indicating that it is an unauthenticated child, the device shall not send a network leave command
8413 frame.

8414 Next, the NWK layer shall issue the NLME-LEAVE.confirm primitive with the DeviceAddress parameter
8415 set to the 64-bit IEEE address of the child device being removed. The Status parameter of the
8416 NLME-LEAVE.confirm primitive shall have a value of SUCCESS if the leave command frame was not
8417 transmitted, *i.e.* in the case of an unauthenticated child. Otherwise, the Status parameter of the
8418 NLME-LEAVE.confirm shall have the same value as the Status parameter returned by the
8419 MCPS-DATA.confirm primitive.

8420 After the child device has been removed, the NWK layer of the parent should modify its neighbor table,
8421 and any other internal data structures that refer to the child device, to indicate that the device is no longer
8422 on the network. It is an error for the next higher layer to address and transmit frames to a child device after
8423 that device has been removed.

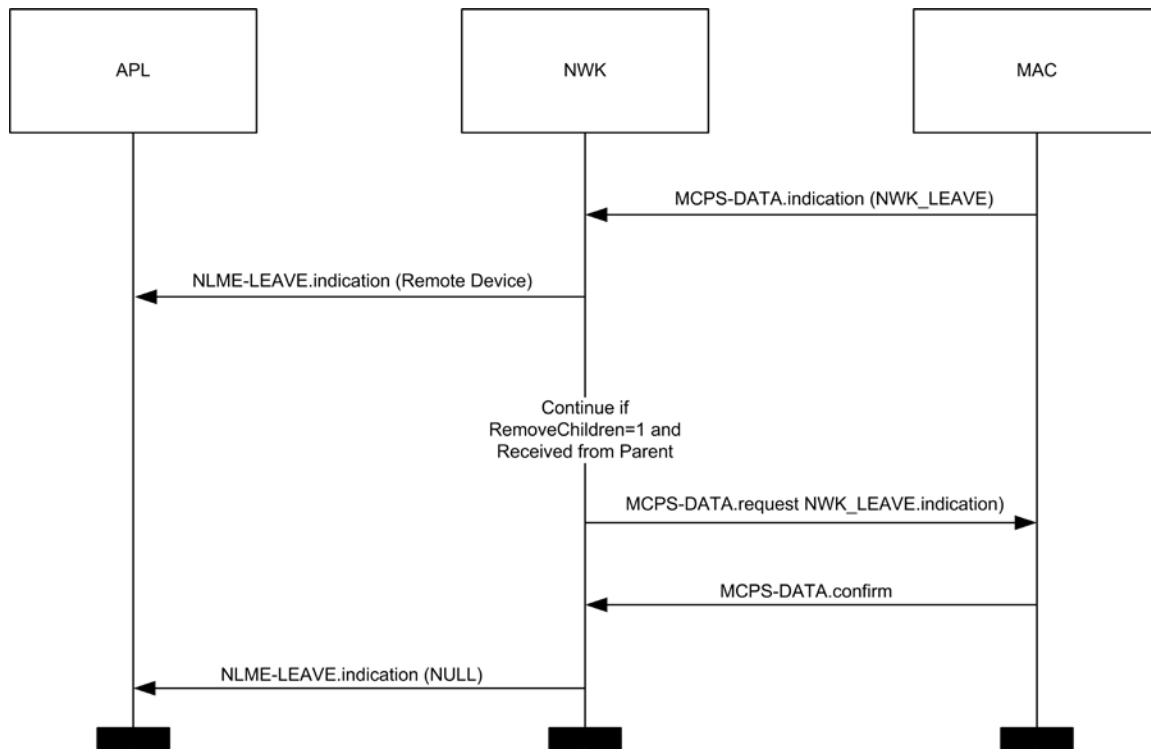
8424 If an unauthenticated child device is removed from the network before it is authenticated, then the address
8425 formerly in use by the device being asked to leave may be assigned to another device that joins subse-
8426 quently.

8427 ZigBee end devices have no child devices to remove and should not receive NLME-LEAVE.request primitives
8428 with non-NULL DeviceAddress parameters.

3.6.1.10.3 Upon Receipt of the Leave Command Frame

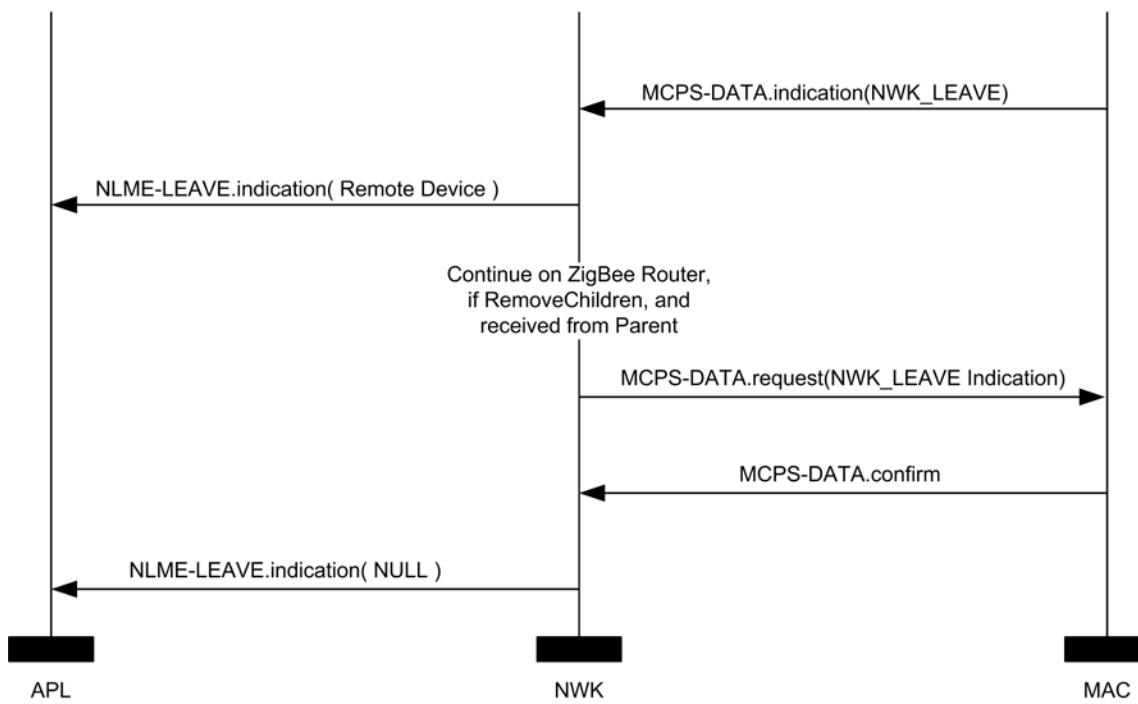
Upon receipt of the leave command frame by the NWK layer via the MCPS-DATA.indication primitive, as shown in Figure 3.46, the device shall check the value of the request sub-field of the command options field of the command frame. If the request sub-field has a value of 0, then the NWK layer shall issue the NLME-LEAVE.indication primitive to the next higher layer with the device address parameter equal to the value in the source IEEE Address sub-field of the leave command frame. The device should also modify its neighbor table, and any other internal data structures that refer to the leaving device, to indicate that the leaving device is no longer on the network. It is an error for the next higher layer to address and transmit frames to a device after that device has left the network.

Figure 3.46 On Receipt of a Leave Command



8441

Figure 3.47 On Receipt of a Leave Command by a ZED



8442

APL

NWK

MAC

8443

8444 If, on receipt by the NWK layer of a ZigBee router of a leave command frame as described above, the
8445 `SrcAddr` parameter of the `MCPS-DATA.indication` that delivered the command frame is the 16-bit network
8446 address of the parent of the recipient, and the value of the remove children sub-field of the command options
8447 field is found to have a value of 1, then the recipient shall send a leave command frame using the
8448 `MCPS-DATA.request` primitive with the `DstAddr` parameter set to `0xffff` indicating a MAC broadcast. The
8449 request sub-field of the command options field of the leave command frame shall be set to 0.

8450 The value of the remove children sub-field and the rejoin sub-field of the command options field of the
8451 outgoing leave command shall reflect the value of the same field for the incoming leave command frame.
8452 After transmission of the leave command frame, it shall set the `nwkExtendedPANId` attribute of the NIB to
8453 `0x0000000000000000` and it shall issue a `NLME-LEAVE.indication` primitive to the higher layer with
8454 `DeviceAddress` parameter equal to `NULL`.

8455 If the request sub-field has a value of 1 then the procedure in section 3.6.1.10.3.1 shall be executed.:¹

3.6.1.10.3.1 Validation of the leave request

8457 The following procedure applies to processing of the NWK Leave (request) command frame and the ZDO
8458 `Mgmt_leave_req`.

- 8459 1. If the device is a ZigBee Coordinator, the message shall be dropped and no further processing
8460 shall be performed.
- 8461 2. If the device is ZigBee Router, the following shall be performed:
 - a. The device shall not consider the Relationship field within the `nwkNeighborTable` entry
8463 corresponding to the sending device.
 - b. If the `nwkLeaveRequestAllowed` in the NIB is TRUE, the device shall perform the pro-
8465 cedure described in 3.6.1.10.1. No further processing is performed.

¹ CCB 1548

8466 c. Otherwise if nwkLeaveRequestAllowed in the NIB is FALSE, no further processing is
8467 performed.

8468 3. If the device is a ZigBee End Device, the following shall be performed:

8469 a. Examine the nwkNeighborTable for an entry where the Network Address is the same as
8470 the SrcAddr parameter of the MCPS-DATA.indication primitive that delivered the NWK
8471 command.

8472 i. If no entry is found, then no further processing shall be done.

8473 b. If the corresponding entry in the nwkNeighborTable has a Relationship value that is not
8474 0x00 (neighbor is the parent), then no further processing shall be done.

8475 c. The sending device is the parent of the receiving device, the receiving device shall per-
8476 form the procedure described in 3.6.1.10.1. No further processing is performed.

8477 4. No further processing is performed.

8478 If a ZigBee end device receives a leave command frame as described above and the SrcAddr parameter of
8479 the MCPS-DATA.indication that delivered the command frame is the 16-bit network address of the parent
8480 of the recipient, it shall set the *nwkExtendedPANId* attribute of the NIB to 0x0000000000000000 and shall
8481 issue a NLME-LEAVE.indication primitive to the higher layer with DeviceAddress parameter equal to
8482 NULL.

8483 The NWK layer may employ retry techniques, as described in section 3.6.5 to enhance the reliability of the
8484 leave procedure but, beyond this note, these mechanisms are outside the scope of this specification.

8486 **3.6.1.10.4 Local Process for Leaving the network**

8487 Upon receipt of a NLME-LEAVE.request primitive or the NWK layer leave command, the following shall
8488 be employed.

8489 1. If the Rejoin value is set to 1 in either the NLME-LEAVE.request primitive or the NWK Leave
8490 command, it shall do the following.

8491 a. The device may execute the rejoin procedure by issuing an NLME-JOIN.request with the
8492 RejoinNetwork set to 1.

8493 b. No further processing shall take place.

8494 2. If the Rejoin value is set to 0, it shall clear the following values in the NIB:

8495 a. nwkNeighborTable

8496 b. nwkRouteTable

8497 c. nwkManagerAddr

8498 d. nwkUpdateId

8499 e. nwkNetworkAddress

8500 f. nwkGroupIDTable

8501 g. nwkExtendedPANID

8502 h. nwkRouteRecordTable

8503 i. nwkIsConcentrator

8504 j. nwkConcentratorRadius

8505 k. nwkSecurityMaterialSet

8506 l. nwkActiveKeySeqNumber

- 8507 m. nwkAddressMap
8508 n. nwkPANID
8509 o. nwkTxTotal
8510 p. nwkParentInformation
8511 3. The device is no longer operating on the network.

8512

8513 **3.6.1.11 Changing the ZigBee Coordinator Configuration**

8514 If the next higher layer of a ZigBee coordinator device wishes to change the configuration of the network, it
8515 shall request that the MAC sub-layer instigate the changes in its PIB. The ZigBee coordinator configuration
8516 is composed of the following items:

- 8517 • Whether or not the device wishes to be a ZigBee parent
8518 • The beacon order of the MAC superframe
8519 • The superframe order of the MAC superframe
8520 • Whether or not battery life extension mode is to be used

8521 A change to the ZigBee coordinator configuration is initiated by issuing the
8522 NLME-NETWORK-FORMATION.request primitive to the NLME. The status of the attempt is communicated
8523 back via the NLME-NETWORK-FORMATION.confirm primitive.

8524 For more details on the impact of such changes imposed on the MAC sub-layer see IEEE 802.15.4-2003
8525 [B1].

8526 **3.6.1.12 Resetting a Device**

8527 The NWK layer of a device shall be reset immediately following initial power-up, before a join attempt to a
8528 new network and after a leave attempt where the device is not intending to rejoin the network. This process
8529 should not be initiated at any other time. A reset is initiated by issuing the NLME-RESET.request primitive
8530 to the NLME and the status of the attempt is communicated back via the NLME-RESET.confirm primitive.
8531 The reset process shall clear the routing table entries of the device.

8532 Some devices may store NWK layer quantities in non-volatile memory and restore them after a reset. The
8533 WarmStart parameter of the NLME-RESET.request may also be used for this purpose. When *nwkAddrAlloc*
8534 is equal to 0x00, a device always gets a network address from its parent upon joining or rejoining. The
8535 new network address may be different from its old network address. In such a case, any device that is
8536 communicating with the device that has been reset must rediscover the device using higher-layer protocols.
8537 When *nwkAddrAlloc* is equal to 0x02, a device may use the same address on rejoining a network and
8538 therefore should not discard its address on reset unless it does not intend to rejoin the same network.

8539 **3.6.1.13 Managing a PANId Conflict**

8540 Since the 16-bit PANID is not a unique number there is a possibility of a PANId conflict. The next section
8541 explains how — through the use of the Network Report and Network Update command frames — the PA-
8542 NId of a network can be updated.

8543 **3.6.1.13.1 Detecting a PANId Conflict**

8544 Any device that is operational on a network and receives an MLME-BEACON-NOTIFY.indication in
8545 which the PAN identifier of the beacon frame matches its own PAN identifier but the EPID value contained
8546 in the beacon payload is either not present or not equal to *nwkExtendedPANID*, shall be considered to have
8547 detected a PAN Identifier conflict.

8548 A node that has detected a PAN identifier conflict shall construct a Network Report Command frame of
8549 type PAN Identifier Conflict which shall be sent to the device identified by the address given in the *nwk-*
8550 *ManagerAddr* attribute of the NIB. The Report Information field will contain a list of all the 16-bit PAN
8551 identifiers that are being used in the local neighborhood. How this list is created is outside the scope of the
8552 specification, however it is recommended that it be constructed from the results of an
8553 MLME-SCAN.request of type ACTIVE.

3.6.1.13.2 Upon Receipt of a Network Report Command Frame

8555 The device identified by the 16-bit network address contained within the *nwkManagerAddr* attribute of the
8556 NIB shall be the recipient of network report command frames of type PAN identifier conflict.

8557 On receipt of the network report command frame, the designated network layer function manager shall se-
8558 lect a new 16-bit PAN identifier for the network. The new PAN identifier is chosen at random, but a check
8559 is performed to ensure that the chosen PAN identifier is not already in use in the local neighborhood and
8560 also not contained within the Report Information field of the network report command frame.

8561 Once a new PAN identifier has been selected, the designated network layer function manager shall first in-
8562 crement the NIB attribute *nwkUpdateId* (wrapping around to 0 if necessary) and then shall construct a net-
8563 work update command frame of type PAN identifier update. The update information field shall be set to the
8564 value of the new PAN identifier. The network update command frame shall be sent to the ZigBee coordi-
8565 nator.

8566 After it sends out this command frame, the designated network layer function manager shall start a timer
8567 with a value equal to *nwkNetworkBroadcastDeliveryTime* OctetDurations. When the timer expires, the
8568 ZigBee coordinator shall change its current PAN ID to the newly selected one by reissuing the
8569 MLME-START.request with the new PANID.

8570 Upon transmission of the Network Update command frame the designated network layer function manager
8571 shall create a NLME-NWK-STATUS.indication primitive with the NetworkAddr parameter set to 0 and the
8572 Status parameter set to PAN Identifier Update.

3.6.1.13.3 Upon Receipt of a Network Update Command Frame

8573 On receipt of a network update command frame of type PAN identifier update, a device shall start a timer
8574 with a value equal to *nwkNetworkBroadcastDeliveryTime* OctetDurations. When the timer expires, the de-
8575 vice shall change its current PAN Identifier to the value contained within the Update Information field.

8576 Upon transmission of the network update command frame the device shall create a
8577 NLME-NWK-STATUS.indication primitive with the NetworkAddr parameter set to 0 and the Status pa-
8578 rameter set to PAN Identifier Update.

8579 Upon receipt of the Network Update command from the device identified by the *nwkManagerAddr* attrib-
8580 ute of the NIB, the value contained in the update id field shall be stored in *nwkUpdateId* attribute in the
8581 NIB. The beacon payload shall also be updated.

3.6.2 Transmission and Reception

3.6.2.1 Transmission

8582 Only those devices that are currently associated shall send data frames from the NWK layer. If a device that
8583 is not associated receives a request to transmit a frame, it shall discard the frame and notify the higher layer
8584 of the error by issuing an NLDE-DATA.confirm primitive with a status of INVALID_REQUEST.

8585 All frames handled by or generated within the NWK layer shall be constructed according to the general
8586 frame format specified in Figure 3.5 and transmitted using the MAC sub-layer data service.

8590 For data frames originating at a higher layer, the value of the source address field MAY be supplied using
8591 the Source address parameter of the NLDE-DATA.request primitive. If a value is not supplied or when the
8592 NWK layer needs to construct a new NWK layer command frame, then the source address field SHALL be
8593 set to the value of the *macShortAddress* attribute in the MAC PIB. Support of this parameter in the
8594 NLDE-DATA.request primitive is required if GP feature is to be supported by the implementation.

8595 In addition to source address and destination address fields, all NWK layer transmissions shall include a
8596 radius field and a sequence number field. For data frames originating at a higher layer, the value of the radius
8597 field may be supplied using the Radius parameter of the NLDE-DATA.request primitive. If a value is
8598 not supplied, then the radius field of the NWK header shall be set to twice the value of the *nwkMaxDepth*
8599 attribute of the NIB (see Constants and NIB Attributes). For data frames originating at a higher layer, the
8600 value of the sequence number field MAY be supplied using the Sequence number parameter of the
8601 NLDE-DATA.request primitive. If a value is not supplied or when the NWK layer needs to construct a new
8602 NWK layer command frame, then the NWK layer SHALL supply the value. Support of this parameter in the
8603 NLDE-DATA.request primitive is required if GP feature is to be supported by the implementation.
8604 The NWK layer on every device shall maintain a sequence number that is initialized with a random value.
8605 The sequence number shall be incremented by 1, each time the NWK layer supplies a new sequence num-
8606 ber value for a NWK frame. The value of the sequence number shall be inserted into the sequence num-
8607 ber field of the frame's NWK header.

8608 Once an NPDU is complete, if security is required for the frame, it shall be passed to the security service
8609 provider for subsequent processing according to the specified security suite (see section 4.2.2). Security
8610 processing is not required if the SecurityEnable parameter of the NLDE-DATA.request is equal to FALSE.
8611 If the NWK security level as specified in *nwkSecurityLevel* is equal to 0, then the security sub-field of the
8612 frame control field shall always be set to 0.

8613 On successful completion of the secure processing, the security suite returns the frame to the NWK layer
8614 for transmission. The processed frame will have the correct auxiliary header attached. If security processing
8615 of the frame fails and the frame was a data frame, the frame will inform the higher layer of the
8616 NLDE-DATA.confirm primitive's status. If security processing of the frame fails and the frame is a net-
8617 work command frame, it is discarded and no further processing shall take place.

8618 When the frame is constructed and ready for transmission, it shall be passed to the MAC data service. An
8619 NPDU transmission is initiated by issuing the MCPS-DATA.request primitive to the MAC sub-layer. The
8620 MCPS-DATA.confirm primitive then returns the results of the transmission.

8621 **3.6.2.2 Reception and Rejection**

8622 In order to receive data, a device must enable its receiver. The next higher layer may initiate reception us-
8623 ing the NLME-SYNC.request primitive. On a beacon-enabled network, receipt of this primitive by the
8624 NWK layer shall cause a device to synchronize with its parent's next beacon and, optionally, to track future
8625 beacons. The NWK layer shall accomplish this by issuing an MLME-SYNC.request to the MAC sub-layer.
8626 On a non-beacon-enabled network, the NLME-SYNC.request shall cause the NWK layer of a device with
8627 *macRxOnWhenIdle* set to FALSE to poll the device's parent using the MLME-POLL.request primitive.

8628 On a non-beacon-enabled network, the NWK layer on a ZigBee coordinator or ZigBee router shall ensure,
8629 to the maximum extent feasible, that the receiver is enabled whenever the device is not transmitting. On a
8630 beacon-enabled network, the NWK layer should ensure that the receiver is enabled when the device is not
8631 transmitting during the active period of its own superframe and of its parent's superframe. The NWK layer
8632 may use the *macRxOnWhenIdle* attribute of the MAC PIB for this purpose.

8633 Once the receiver is enabled, the NWK layer will begin to receive frames via the MAC data service. On
8634 receipt of each frame, the radius field of the NWK header shall be decremented by 1. If, as a result of being
8635 decremented, this value falls to 0, the frame shall not, under any circumstances, be retransmitted. It may,
8636 however, be passed to the next higher layer or otherwise processed by the NWK layer as outlined else-
8637 where in this specification.

8638 The NWK layer SHALL accept non-incremental NWK-level values in the Sequence number field of the
8639 ZigBee Network header for consecutive packets with the same value of the Source address field of the
8640 ZigBee Network header.

8641 On receipt of a frame with the End Device Initiator sub-field of the frame control set to 1, the following
8642 processing shall take place.

- 8643 1. If the receiving device is an end device the message shall be dropped and no further processing
8644 shall take place.
- 8645 2. The receiving device shall search the neighbor table for an entry where the value of the Network
8646 Address matches the value of the Source Address field of the message, and the device type is 0x02
8647 (end device). If no entry is found then the message shall be dropped and no further processing
8648 shall take place.

8649

8650 The following data frames shall be passed to the next higher layer using the NLDE-DATA.indication primitive:

- 8652 • Frames with a broadcast address that matches a broadcast group of which the device is a member.
- 8653 • Unicast data frames and source-addressed data frames for which the destination address matches the
8654 device's network address.
- 8655 • Multicast data frames whose group identifier is listed in the *nwkGroupIDTable*.

8656 If the receiving device is a ZigBee coordinator or an operating ZigBee router, that is, a router that has already invoked the NLME-START-ROUTER.request primitive, it shall process data frames as follows:

- 8658 • Messages shall be verified to determine if an end device has switched router parents. This is outlined
8659 in section 3.6.2.3
- 8660 • Broadcast and multicast data frames shall be relayed according to the procedures outlined in sections
8661 3.6.5 and 3.6.6.
- 8662 • Unicast data frames with a destination address that does not match the device's network address shall
8663 be relayed according to the procedures outlined in section 3.6.3.3. (Under all other circumstances,
8664 unicast data frames shall be discarded immediately.)
- 8665 • Source-routed data frames with a destination address that does not match the device's network address
8666 shall be relayed according to the procedures outlined in section 3.6.3.3.2.
- 8667 • The procedure for handling route request command frames is outlined in section 3.6.3.5.2.
- 8668 • The procedure for handling route reply command frames for which the destination address matches the
8669 device's network address is outlined in section 3.6.3.5.3.
- 8670 • Route reply command frames for which the destination address does not match the device's network
8671 address shall be discarded immediately. Network status command frames shall be handled in the same
8672 manner as data frames.

8673 The NWK layer shall indicate the receipt of a data frame to the next higher layer using the
8674 NLDE-DATA.indication primitive.

8675 On receipt of a frame, the NLDE shall check the value of the security sub-field of the frame control field. If
8676 this value is non-zero, the NLDE shall pass the frame to the security service provider (see section 4.2.2) for
8677 subsequent processing according to the specified security suite. If the security sub-field is set to 0, the
8678 *nwkSecurityLevel* attribute in the NIB is non-zero, the device is currently joined and authenticated, and the
8679 incoming frame is a NWK data frame, the NLDE shall discard the frame. If the security sub-field is set to
8680 0, the *nwkSecurityLevel* attribute in the NIB is non-zero, and the incoming frame is a NWK command
8681 frame and the command ID is 0x06 (rejoin request), the NLDE shall only accept the frame if it is destined
8682 to itself, that is, if it does not need to be forwarded to another device. Otherwise the frame shall be
8683 dropped and no further processing done.

8684 If the device is not joined and authenticated, or undergoing the Trust Center Rejoin process, it shall perform
8685 the following checks. If the frame is a NWK command where the security sub-field of the frame is set to
8686 zero then it shall only accept the frame if the command ID is 0x07 (rejoin response). If the frame is a
8687 NWK data frame where the security sub-field is set to 0, the device shall further examine the APDU and
8688 determine if it contains an APS command ID of 0x05 (Transport Key). If the message does not contain an
8689 APS Command of 0x05 (Transport Key), then the message shall be dropped and no further processing
8690 done. All other messages where the security sub-field is set to 0 shall be dropped and no further pro-
8691 cessing shall be done.²

8692 **3.6.2.3 Examination for End Devices that have changed Router 8693 Parents**

8694 A router upon receipt of a NWK command or data message must perform the following:

- 8695 1. Search the neighbor table for an entry where the Network Address matches the value of the NWK
8696 Source field in the message. If no match is found then go to step 6.
- 8697 2. Examine if the Device Type of the entry corresponds to a ZigBee End Device. If it does not, go
8698 to step 6.
- 8699 3. Examine if the MAC source field of the message matches the NWK source field. If it does go to
8700 step 6.
- 8701 4. If the message is a broadcast, examine if an entry exists in nwkBroadcastTransactionTable, if it
8702 does then go to step 6. If the message is a unicast, continue processing.
- 8703 5. At this point the message indicates it has been relayed by another device on the network acting as
8704 the end device's router parent; delete the corresponding neighbor table entry.
- 8705 6. Continue to process the message.

8706 When an end device joins or rejoins it will broadcast a ZDO Device_ance, which in turn will be processed
8707 as follows:

- 8708 1. Search the neighbor table for an entry where the IEEEAddr in the ZDO Device_ance command
8709 frame matches the Extended Address field of the neighbor table entry and the Device Type field in
8710 the neighbor table entry is equal to End Device (0x02).
- 8711 2. If no such entry is found, skip to step 4.
- 8712 3. If an entry is found and the Device_Ance was broadcast, examine the nwkBroadcastTransac-
8713 tionTable. If there is no entry in the nwkBroadcastTransactionTable for this message, this indi-
8714 cates the message was relayed by another device on the network acting as the end device's router
8715 parent.
 - 8716 a. Delete the neighbor table entry with the corresponding Extended Address equal to the
8717 IEEEAddr in the Device_Ance command.
- 8718 4. Continue processing the Device_Ance message.

8720 **3.6.3 Routing**

8721 ZigBee coordinators and routers shall provide the following functionality:

- 8722 • Relay data frames on behalf of higher layers
- 8723 • Relay data frames on behalf of other ZigBee routers
- 8724 • Participate in route discovery in order to establish routes for subsequent data frames

² CCB 1941

- 8725 • Participate in route discovery on behalf of end devices
- 8726 • Participate in route repair
- 8727 • Employ the ZigBee path cost metric as specified in route discovery
- 8728 ZigBee coordinators or routers may provide the following functionality:
- 8729 • Maintain routing tables in order to remember best available routes
- 8730 • Initiate route discovery on behalf of higher layers
- 8731 • Initiate route discovery on behalf of other ZigBee routers
- 8732 • Initiate route repair
- 8733 • Conduct neighbor routing

8734 **3.6.3.1 Routing Cost**

8735 The ZigBee routing algorithm uses a path cost metric for route comparison during route discovery and
8736 maintenance. In order to compute this metric, a cost, known as the link cost, is associated with each link in
8737 the path and link cost values are summed to produce the cost for the path as a whole.

8738 More formally, if we define a path P of length L as an ordered set of devices and a link, as a sub-path of
8739 length 2, then the path cost

$$8740 \quad C\{P\} = \sum_{i=1}^{L-1} C\{[D_i, D_{i+1}]\}$$

8742 where each of the values is referred to as a link cost. The link cost for a link l is a function with values in
8743 the interval defined as:

$$8744 \quad C\{l\} = \begin{cases} 7, \\ \min\left(7, \text{round}\left(\frac{1}{p_l^4}\right)\right) \end{cases}$$

8745 where p_l is defined as the probability of packet delivery on the link l .

8746 Thus, implementers may report a constant value of 7 for link cost or they may report a value that reflects
8747 the probability p_l of reception — specifically, the reciprocal of that probability — which should, in turn,
8748 reflect the number of expected attempts required to get a packet through on that link each time it is used. A
8749 device that offers both options may be forced to report a constant link cost by setting the value of the NIB
8750 attribute *nwkReportConstantCost* to TRUE. If the *nwkSymLink* attribute of the NIB has a value of TRUE,
8751 then the *nwkReportConstantCost* attribute must have a value of FALSE, and the NWK layer must calculate
8752 routing cost in the manner described above.

8753 The question that remains, however, is how p_l is to be estimated or measured. This is primarily an imple-
8754 mentation issue, and implementers are free to apply their ingenuity. p_l may be estimated over time by actu-
8755 ally counting received beacon and data frames and observing the appropriate sequence numbers to detect
8756 lost frames. This is generally regarded as the most accurate measure of reception probability. However, the
8757 most straightforward method, available to all, is to form estimates based on an average over the per-frame
8758 LQI value provided by the IEEE 802.15.4-2003 MAC and PHY. Even if some other method is used, the ini-
8759 tial cost estimates shall be based on average LQI. A table-driven function may be used to map average LQI
8760 values onto $C\{l\}$ values. It is strongly recommended that implementers check their tables against data de-
8761 rived from tests performed on production hardware, as inaccurate costs will hamper the operating ability of
8762 the ZigBee routing algorithm.

3.6.3.2 Routing Tables

A ZigBee router or ZigBee coordinator may maintain a routing table. The information that shall be stored in a ZigBee routing table entry is shown in Table 3.56. The aging and retirement of routing table entries in order to reclaim table space from entries that are no longer in use is a recommended practice; it is, however, out of scope of this specification.

Table 3.56 Routing Table Entry

Field Name	Size	Description
Destination address	2 octets	The 16-bit network address or Group ID of this route. If the destination device is a ZigBee router, ZigBee coordinator, or an end device, and <i>nwkAddrAlloc</i> has a value of 0x02, this field shall contain the actual 16-bit address of that device. If the destination device is an end device and <i>nwkAddrAlloc</i> has a value of 0x00, this field shall contain the 16-bit network address of the device's parent.
Status	3 bits	The status of the route. See Table 3.57 for values.
No route cache	1 bit	A flag indicating that the destination indicated by this address does not store source routes.
Many-to-one	1 bit	A flag indicating that the destination is a concentrator that issued a many-to-one route request.
Route record required	1 bit	A flag indicating that a route record command frame should be sent to the destination prior to the next data packet.
GroupID flag	1 bit	A flag indicating that the destination address is a Group ID.
Next-hop address	2 octets	The 16-bit network address of the next hop on the way to the destination.

Table 3.57 enumerates the values for the route status field.

Table 3.57 Route Status Values

Numeric Value	Status
0x0	ACTIVE
0x1	DISCOVERY_UNDERWAY
0x2	DISCOVERY_FAILED
0x3	INACTIVE
0x4	VALIDATION_UNDERWAY

0x5 – 0x7	Reserved
-----------	----------

8772

8773 This section describes the routing algorithm. The term “routing table capacity” is used to describe a situation
8774 in which a device has the ability to use its routing table to establish a route to a particular destination
8775 device. A device is said to have routing table capacity if:

- 8776 • It is a ZigBee coordinator or ZigBee router
8777 • It maintains a routing table
8778 • It has a free routing table entry or it already has a routing table entry corresponding to the destination

8779 If a ZigBee router or ZigBee coordinator maintains a routing table, it shall also maintain a route discovery
8780 table containing the information shown in Table 3.58. Routing table entries are long-lived, while route dis-
8781 covery table entries last only as long as the duration of a single route discovery operation and may be re-
8782 used.

8783 **Table 3.58 Route Discovery Table Entry**

Field Name	Size (octets)	Description
Route request ID	1	A sequence number for a route request command frame that is incremented each time a device initiates a route request.
Source address	2	The 16-bit network address of the route request’s initiator.
Sender address	2	The 16-bit network address of the device that has sent the most recent lowest cost route request command frame corresponding to this entry’s route request identifier and source address. This field is used to determine the path that an eventual route reply command frame should follow.
Forward cost	1	The accumulated path cost from the source of the route request to the current device.
Residual cost	1	The accumulated path cost from the current device to the destination device.
Expiration time	2	A countdown timer indicating the number of milliseconds until route discovery expires. The initial value is <i>nwkcRouteDiscoveryTime</i> .

8784

8785 A device is said to have “route discovery table capacity” if:

- 8786 • It maintains a route discovery table
8787 • It has a free entry in its route discovery table

8788 If a device has both routing table capacity and route discovery table capacity then it may be said to have
8789 “routing capacity.”

8790 During route discovery, the information that a ZigBee router or ZigBee coordinator is required to maintain
8791 in order participate in the discovery of a particular route is distributed between a routing table entry and a
8792 route discovery table entry. Once discovery has been completed, only the routing table entry need be main-
8793 tained in order for the NWK layer to perform routing along the discovered route. Throughout this section,
8794 references are made to this relationship between a routing table entry and its “corresponding” route discov-
8795 ery table entry and vice versa. The maintenance of this correspondence is up to the implementer since en-
8796 tries in the tables have no elements in common, but it is worth noting in this regard that the unique “keys”
8797 that define a route discovery are the source address of the route discovery command frame and the route
8798 request ID generated by that device and carried in the command frame payload.

8799 If a device has the capability to initiate a many-to-one route request, it may also maintain a route record ta-
8800 ble (see Table 3.50).

8801 **3.6.3.3 Upon Receipt of a Unicast Frame**

8802 On receipt of a unicast frame from the MAC sub-layer, or an NLDE-DATA.request from the next higher
8803 layer, the NWK layer routes it according to the following procedure.

8804 If the receiving device is a ZigBee router or ZigBee coordinator, and the destination of the frame is a
8805 ZigBee end device and also the child of the receiving device, the frame shall be routed directly to the des-
8806 tination using the MCPS-DATA.request primitive, as described in section 3.6.2.1. The frame shall also set
8807 the next hop destination address equal to the final destination address. Otherwise, for purposes of the ensu-
8808 ing discussion, define the *routing address* of a device to be its network address if it is a router or the coordi-
8809 nator or an end device and *nwkAddrAlloc* has a value of 0x02, or the network address of its parent if it is
8810 an end device and *nwkAddrAlloc* has a value of 0x00. Define the *routing destination* of a frame to be the
8811 routing address of the frame’s NWK destination. Note that distributed address assignment makes it possible
8812 to derive the routing address of any device from its address. See section 3.6.1.6 for details.

8813 A ZigBee router or ZigBee coordinator may check the neighbor table for an entry corresponding to the
8814 routing destination of the frame. If there is such an entry, the device may route the frame directly to the
8815 destination using the MCPS-DATA.request primitive as described in section 3.6.2.1.

8816 A device that has routing capacity shall check its routing table for an entry corresponding to the routing
8817 destination of the frame. If there is such an entry, and if the value of the route status field for that entry is
8818 ACTIVE or VALIDATION_UNDERWAY, the device shall relay the frame using the
8819 MCPS-DATA.request primitive and set the route status field of that entry to ACTIVE if it does not already
8820 have that value. If the many-to-one field of the routing table entry is set to TRUE, the NWK shall follow
8821 the procedure outlined in section 3.6.3.5.4 to determine whether a route record command frame must be
8822 sent.

8823 When relaying a unicast frame, the SrcAddrMode and DstAddrMode parameters of the
8824 MCPS-DATA.request primitive shall both have a value of 0x02, indicating 16-bit addressing. The SrcPA-
8825 NIId and DstPANId parameters shall both have the value provided by the macPANId attribute of the MAC
8826 PIB for the relaying device. The SrcAddr parameter shall be set to the value of *macShortAddress* from the
8827 MAC PIB of the relaying device, and the DstAddr parameter shall be the value provided by the next-hop
8828 address field of the routing table entry corresponding to the routing destination. Bit *b0* of the TxOptions
8829 parameter should be set to 1, indicating acknowledged transmission.

8830 The NWK Sequence Number of a replayed packet shall not be changed by a router device relaying the
8831 packet. The router device relaying a packet shall leave the NWK Sequence Number of the originating de-
8832 vice in the NWK Sequence Number field.

8833 If the device has a routing table entry corresponding to the routing destination of the frame but the value of
8834 the route status field for that entry is DISCOVERY_UNDERWAY, the device shall determine if it initiated
8835 the discovery by consulting its discovery table. If the device initiated the discovery, the frame shall be
8836 treated as though route discovery has been initiated for this frame, otherwise, the device shall initiate route
8837 discovery as described in section 3.6.3.5.1. The frame may optionally be buffered pending route discovery
8838 or routed along the tree using hierarchical routing, provided that the NIB attribute *nwkUseTreeRouting* has
8839 a value of TRUE. If the frame is routed along the tree, the discover route sub-field of the NWK header
8840 frame control field shall be set to 0x00.

8841 If the device has a routing table entry corresponding to the routing destination of the frame but the route
8842 status field for that entry has a value of DISCOVERY_FAILED or INACTIVE, the device may route the
8843 frame along the tree using hierarchical routing, provided that the NIB attribute *nwkUseTreeRouting* has a
8844 value of TRUE. If the device does not have a routing table entry for the routing destination with a status
8845 value of ACTIVE or VALIDATION_UNDERWAY, and it received the frame from the next higher layer,
8846 it shall check its source route table for an entry corresponding to the routing destination. If such an entry is
8847 found and the length is less than *nwkMaxSourceRoute*, the device shall transmit the frame using source
8848 routing as described in section 3.6.3.3.1. If the device does not have a routing table entry for the routing
8849 destination and it is not originating the frame using source routing, it shall examine the discover route
8850 sub-field of the NWK header frame control field. If the discover route sub-field has a value of 0x01, the
8851 device shall initiate route discovery, as described in section 3.6.3.5.1. If the discover route sub-field has a
8852 value of 0 and the NIB attribute *nwkUseTreeRouting* has a value of TRUE then the device shall route along
8853 the tree using hierarchical routing. If the discover route sub-field has a value of 0, the NIB attribute
8854 *nwkUseTreeRouting* has a value of FALSE, and there is no routing table corresponding to the routing des-
8855 tination of the frame, the frame shall be discarded and the NLDE shall issue the NLDE-DATA.confirm
8856 primitive with a status value of ROUTE_ERROR.

8857 A device without routing capacity shall route along the tree using hierarchical routing provided that the
8858 value of the NIB attribute *nwkUseTreeRouting* is TRUE. If the value of the NIB attribute *nwkUs-
8859 eTreeRouting* is FALSE, the frame shall be discarded. If the frame is the result of an NLDE-DATA.request
8860 from the NHL of the current device, the NLDE shall issue the NLDE-DATA.confirm primitive with a sta-
8861 tus value of ROUTE_ERROR. If the frame is being relayed on behalf of another device, the NLME shall
8862 issue a network status command frame destined for the device that is the source of the frame with a status
8863 of 0x04, indicating a lack of routing capacity. It shall also issue the NLME-NWK-STATUS.indication to
8864 the next higher layer with the NetworkAddr parameter equal to the 16-bit network address of the frame,
8865 and the Status parameter equal to 0x04, indicating a lack of routing capacity.

8866 For hierarchical routing, if the destination is a descendant of the device, the device shall route the frame to
8867 the appropriate child. If the destination is a child, and it is also an end device, delivery of the frame can fail
8868 due to the *macRxOnWhenIdle* state of the child device. If the child has *macRxOnWhenIdle* set to FALSE,
8869 indirect transmission as described in IEEE 802.15.4-2003 [B1] may be used to deliver the frame. If the des-
8870 tination is not a descendant, the device shall route the frame to its parent.

8871 Every other device in the network is a descendant of the ZigBee coordinator and no device in the network
8872 is the descendant of any ZigBee end device. For a ZigBee router with address *A* at depth *d*, if the following
8873 logical expression is true, then a destination device with address *D* is a descendant:

8874
$$A < D < A + Cskip(d - 1)$$

8875 For a definition of *Cskip(d)*, see section 3.6.1.6.

8876 If it is determined that the destination is a descendant of the receiving device, the address *N* of the next hop
8877 device is given by:

8878
$$N = D$$

8879 for ZigBee end devices, where $D > A + Rm \times Cskip(d)$, and otherwise:

8880
$$N = A + 1 + \left\lfloor \frac{D - (A + 1)}{Cskip(d)} \right\rfloor \times Cskip(d)$$

If the NWK layer on a ZigBee router or ZigBee coordinator fails to deliver a unicast or multicast frame for any reason, the router or coordinator shall make its best effort to report the failure. No failure should be reported as the result of a failure to deliver a NLME-NWK-STATUS. The failure reporting may take one of two forms. If the failed frame was being relayed as a result of a request from the next higher layer, then the NWK layer shall issue an NLDE-DATA.confirm with the error to the next higher layer. The value of the NetworkAddr parameter of the primitive shall be the intended destination of the frame. If the frame was being relayed on behalf of another device, then the relaying device shall send a network status command frame back to the source of the frame. The destination address field of the network status command frame shall be taken from the destination address field of the failed data frame.

In either case, the reasons for failure that may be reported appear in Table 3.42.

3.6.3.3.1 Originating a Source Routed Data Frame

If, on receipt of a data frame from the next higher layer, it is determined that the frame should be transmitted using source routing as described above, the source route shall be retrieved from the route record table.

If there are no intermediate relay nodes, the frame shall be transmitted directly to the routing destination without source routing by using the MCPS-DATA.request primitive, with the DstAddr parameter value indicating the routing destination.

If there is at least one relay node, the source route flag of the NWK header frame control field shall be set, and the NWK header source route subframe shall be present. The relay count sub-field of the source route subframe shall have a value equal to the number of relays in the relay list. The relay index sub-field shall have a value equal to 1 less than the number of relays. The relay list sub-field shall contain the list of relay addresses, least significant octet first. The relay closest to the destination shall be listed first. The relay closest to the originator shall be listed last.

The device shall relay the frame using the MCPS-DATA.request primitive. The DstAddr parameter shall have the value of the final relay address in the relay list.

3.6.3.3.2 Relaying a Source Routed Data Frame

Upon receipt of a source routed data frame from the MAC sub-layer as described in section 3.6.3.3, if the relay index sub-field of the source route sub-frame has a value of 0, the device shall check the destination address field of the NWK header of the frame. If the destination address field of the NWK header of the frame is equal in value to the *nwkNetworkAddress* attribute of the NIB, then the frame shall be passed to the next higher layer using the NLDE-DATA.indication primitive. If the destination address field is not equal to the *nwkNetworkAddress* attribute of the NIB, and the receiving device is a ZigBee router or ZigBee coordinator, the device shall relay the frame directly to the NWK header destination using the MCPS-DATA.request primitive, otherwise the frame shall be discarded silently.

If the relay index sub-field has a value other than 0, the device shall compare its network address with the address found at the relay index in the relay list. If the addresses do not match, the frame shall be discarded and no further action shall be taken. Otherwise, as long as the destination address is not the address of an end device where the relaying device is the parent, the device shall decrement the relay index sub-field by 1, and relay the frame to the address immediately prior to its own address in the relay list sub-field. If the destination address of the frame is an end device child of the relaying device, the frame shall be unicast using the MCPS-DATA.request primitive.

When relaying a source routed data frame, the NWK layer of a device shall also examine the routing table entry corresponding to the source address of the frame. If the no route cache field of the routing table entry has a value of FALSE, then the route record required field of the routing table entry shall be set to FALSE.

3.6.3.4 Link Status Messages

Wireless links may be asymmetric, that is, they may work well in one direction but not the other. This can cause route replies to fail, since they travel backwards along the links discovered by the route request.

8927 For many-to-one routing and two-way route discovery (*nwkSymLink* = TRUE), it is a requirement to dis-
8928 cover routes that are reliable in both directions. To accomplish this, routers exchange link cost measure-
8929 ments with their neighbors by periodically transmitting link status frames as a one-hop broadcast. The re-
8930 verse link cost information is then used during route discovery to ensure that discovered routes use
8931 high-quality links in both directions.

8932 **3.6.3.4.1 Initiation of a Link Status Command Frame**

8933 When joined to a network, a ZigBee router or coordinator shall periodically send a link status command
8934 every *nwkLinkStatusPeriod* seconds, as a one-hop broadcast without retries. It may be sent more frequently
8935 if desired. Random jitter should be added to avoid synchronization with other nodes. See section 3.4.8 for
8936 the link status command frame format.

8937 End devices do not send link status command frames.

8938 **3.6.3.4.2 Upon Receipt of a Link Status Command Frame**

8939 Upon receipt of a link status command frame by a ZigBee router or coordinator, the age field of the neigh-
8940 bor table entry corresponding to the transmitting device is reset to 0. The list of addresses covered by a
8941 frame is determined from the first and last addresses in the link status list, and the first frame and last frame
8942 bits of the command options field. If the receiver's network address is outside the range covered by the
8943 frame, the frame is discarded and processing is terminated. If the receiver's network address falls within the
8944 range covered by the frame, then the link status list is searched. If the receiver's address is found, the out-
8945 going cost field of the neighbor table entry corresponding to the sender is set to the incoming cost value of
8946 the link status entry. If the receiver's address is not found, the outgoing cost field is set to 0.

8947 End devices do not process link status command frames.

8948 **3.6.3.4.3 Aging the Neighbor Table**

8949 For devices using link status messages, the age fields for routers in the neighbor table are incremented eve-
8950 ry *nwkLinkStatusPeriod*. If the value exceeds *nwkRouterAgeLimit*, the outgoing cost field of the neighbor
8951 table entry is set to 0. In other words, if a device fails to receive *nwkRouterAgeLimit* link status messages
8952 from a router neighbor in a row, the old outgoing cost information is discarded. In this case, the neighbor
8953 entry is considered stale and may be reused if necessary to make room for a new neighbor. End devices do
8954 not issue link status messages and therefore should never be considered stale.

8955 If *nwkAddrAlloc* has a value of 0x00, neighbor table entries for relatives should not be considered stale and
8956 reused.

8957 **3.6.3.5 Route Discovery**

8958 Route discovery is the procedure whereby network devices cooperate to find and establish routes through
8959 the NWK. *Unicast route discovery* is always performed with regard to a particular source device and a par-
8960 ticular destination device. *Multicast route discovery* is performed with respect to a particular source device
8961 and a multicast group. *Many-to-one route discovery* is performed by a source device to establish routes to
8962 itself from all ZigBee routers and ZigBee coordinator, within a given radius. A source device that initiates a
8963 many-to-one route discovery is designated as a concentrator and referred to as such in this document.
8964 Throughout section 3.6.3.5 a *destination address* may be a 16-bit broadcast address, the 16-bit network ad-
8965 dress of a particular device, or a 16-bit multicast address, also known as a multicast group ID. A route re-
8966 quest command whose destination address is the routing address of a particular device and whose route re-
8967 quest option field does not have the multicast bit set, is a *unicast route request*. A route request command
8968 whose route request option field has the multicast bit set is a *multicast route request*. The destination ad-
8969 dress field of a multicast route request shall be a multicast group ID. A route request command payload
8970 whose destination address sub-field is a broadcast address (see Table 3.59) is a *many-to-one route request*.
8971 The multicast bit in the route request option field of a many-to-one route request shall be set to 0.

8972 Note that on RREP new frames shall be created at every hop. In all other cases the packets shall not be not
8973 considered a “new” frame. A new frame shall be one with a new route request identifier. For RREP the se-
8974 quence number is regenerated every hop. For RREC the sequence number does not change with every hop.

8975 **3.6.3.5.1 Initiation of Route Discovery**

8976 The unicast route discovery procedure for a device shall be initiated on a ZigBee router or ZigBee coordinator by the NWK layer up on receipt of an NLME-ROUTE-DISCOVERY.request primitive from the next higher layer where the DstAddrMode parameter has a value of 0x02. Or, up on receipt of an NLDE-DATA.request primitive from a higher layer with the DstAddrMode set to 0x02 and the discover route sub-field set to 0x01, for which there is no routing table entry corresponding to the routing address of the destination device (the 16-bit network address indicated by the DstAddr parameter). Or, on receipt of a frame from the MAC sub-layer for which the value of the destination address field in the NWK header is not the address of the current device, the address of an end device child of the current device, or a broadcast address and:

- 8985 • The discover route sub-field of the frame control field has a value of 0x01, and
- 8986 • there is no routing table entry corresponding to the routing destination of the frame, and
- 8987 • either the value of the source address field of the NWK header of the received frame is the same as the 16-bit network address of one of the end device children of the current device, or
- 8989 • the *nwkUseTreeRouting* attribute of the NIB has a value of TRUE.

8990 The route discovery procedure for a multicast address shall be initiated by the NWK layer either in response to the receipt of an NLME-ROUTE-DISCOVERY.request primitive from the next higher layer where the DstAddrMode parameter has a value of 0x01, or as specified in section 3.6.6.2.2.

8993 If the device initiating route discovery has no routing table entry corresponding to the routing address of the destination device, it shall establish a routing table entry with status equal to DISCOVERY_UNDERWAY. If the device has an existing routing table entry corresponding to the routing address of the destination with status equal to ACTIVE or VALIDATION_UNDERWAY, that entry shall be used and the status field of that entry shall retain its current value. If it has an existing routing table entry with a status value other than ACTIVE or VALIDATION_UNDERWAY, that entry shall be used and the status of that entry shall be set to DISCOVERY_UNDERWAY. The device shall also establish the corresponding route discovery table entry if one with the same initiator and route request ID does not already exist.

9001 Each device issuing a route request command frame shall maintain a counter used to generate route request identifiers. When a new route request command frame is created, the route request counter is incremented and the value is stored in the device's route discovery table in the Route request identifier field. Other fields in the routing table and route discovery table are set as described in section 3.6.3.2.

9005 The NWK layer may choose to buffer the received frame pending route discovery or, if the frame is a unicast frame and the NIB attribute *nwkUseTreeRouting* has a value of TRUE, set the discover route sub-field of the frame control field in the NWK header to 0 and forward the data frame along the tree.

9008 Once the device creates the route discovery table and routing table entries, the route request command frame shall be created with the payload depicted in Figure 3.12. The individual fields are populated as follows:

- 9011 • The command frame identifier field shall be set to indicate the command frame is a route request, see Table 3.40.
- 9013 • The Route request identifier field shall be set to the value stored in the route discovery table entry.
- 9014 • The multicast flag and destination address fields shall be set in accordance with the destination address for which the route is to be discovered.
- 9015 • The path cost field shall be set to 0.

9017 Once created, the route request command frame is ready for broadcast and is passed to the MAC sub-layer using the MCPS-DATA.request primitive.

9019 When broadcasting a route request command frame at the initiation of route discovery, the NWK layer shall retry the broadcast *nwkInitialRREQRetries* times after the initial broadcast, resulting in a maximum of *nwkInitialRREQRetries* + 1 transmissions. The retries will be separated by a time interval of *nwk-cRREQRetryInterval* OctetDurations.

The many-to-one route discovery procedure shall be initiated by the NWK layer of a ZigBee router or coordinator on receipt of an NLME-ROUTE-DISCOVERY.request primitive from the next higher layer where the DstAddrMode parameter has a value of 0x00. A many-to-one route request command frame is not retried; however, a discovery table entry is still created to provide loop detection during the *nwkRouteDiscoveryTime* period. If the NoRouteCache parameter of the NLME-ROUTE-DISCOVERY.request primitive is TRUE, the many-to-one sub-field of the command options field of the command frame payload shall be set to 2. Otherwise, the many-to-one sub-field shall be set to 1. Note that in this case, the NWK layer should maintain a route record table. The destination address field of the NWK header shall be set to 0xffff, the all-router broadcast address. The broadcast radius shall be set to the value in *nwkConcentratorRadius*. A source device that initiates a many-to-one route discovery is designated as a concentrator and referred to as such in this document and the NIB attribute *nwkIsConcentrator* should be set to TRUE. If a device has *nwkIsConcentrator* equal to TRUE and there is a non-zero value in *nwkConcentratorDiscoveryTime*, the network layer should issue a route request command frame each *nwkConcentratorDiscoveryTime*.

3.6.3.5.2 Upon Receipt of a Route Request Command Frame

Upon receipt of a route request command frame, if the device is an end device, it shall drop the frame. Otherwise, it shall determine if it has routing capacity.

If the device does not have routing capacity and the route request is a multicast route request or a many-to-one-route request, the route request shall be discarded and route request processing shall be terminated.

If *nwkAddrAlloc* is 0x00 and the device does not have routing capacity and the route request is a unicast route request, the device shall check if the frame was received along a valid path. A path is valid if the frame was received from one of the device's children and the source device is a descendant of that child device, or if the frame was received from the device's parent device and the source device is not a descendant of the device. If the route request command frame was not received along a valid path, it shall be discarded. Otherwise, the device shall check if it is the intended destination. It shall also check if the destination of the command frame is one of its end device children by comparing the destination address field of the route request command frame payload with the address of each of its end device children, if any. If either the device or one of its end device children is the destination of the route request command frame, it shall reply with a route reply command frame. When replying to a route request with a route reply command frame, the device shall construct a frame with the frame type field set to 0x01. The route reply's source address shall be set to the 16-bit network address of the device creating the route reply and the destination address shall be set to the calculated next hop address, considering the originator of the route request as the destination. The link cost from the next hop device to the current device shall be computed as described in section 3.6.3.1 and inserted into the path cost field of the route reply command frame. The route reply command frame shall be unicast to the next hop device by issuing an MCPS-DATA.request primitive.

If the device is not the destination of the route request command frame, the device shall compute the link cost from the previous device that transmitted the frame, as described in section 3.6.3.1. This value shall be added to the path cost value stored in the route request command frame. The route request command frame shall then be unicast towards the destination using the MCPS-DATA.request service primitive. The next hop for this unicast transmission is determined in the same manner as if the frame were a data frame addressed to the device identified by the destination address field in the payload.

If the device does have routing capacity and the received request is a unicast route request, the device shall check if it is the destination of the command frame by comparing the destination address field of the route request command frame payload with its own address. It shall also check if the destination of the command frame is one of its end device children by comparing the destination address field of the route request command frame payload with the address of each of its end device children, if any. If neither the device nor one of its end device children is the destination of the route request command frame, the device shall determine if a route discovery table (see Table 3.58) entry exists with the same route request identifier and source address field. If no such entry exists, one shall be created.

9074 If the device does have routing capacity and the multicast sub-field of the route request command options
9075 field of the received route request frame indicates a multicast route request, the device shall determine
9076 whether an entry already exists in the *nwkGroupIDTable* for which the group identifier field matches the
9077 destination address field of the frame. If a matching entry is found, the device shall determine if a route
9078 discovery table (see Table 3.58) entry exists with the same route request identifier and source address field.
9079 If no such entry exists, one shall be created.

9080 For many-to-one route requests, and for regular route requests if the *nwkSymLink* attribute is TRUE, upon
9081 receipt of a route request command frame, the neighbor table is searched for an entry corresponding to the
9082 transmitting device. If no such entry is found, or if the outgoing cost field of the entry has a value of 0, the
9083 frame is discarded and route request processing is terminated. The maximum of the incoming and outgoing
9084 costs for the neighbor is used for the purposes of the path cost calculation, instead of the incoming cost.
9085 This includes the value used to increment the path cost field of the route request frame prior to retransmission.
9086

9087 When creating the route discovery table entry, the fields are set to the corresponding values in the route
9088 request command frame. The only exception is the forward cost field, which is determined by using the previous
9089 sender of the command frame to compute the link cost, as described in section 3.6.3.1, and adding it to the path cost
9090 contained the route request command frame. The result of the above calculation is stored in the forward cost field
9091 of the newly created route discovery table entry. If the *nwkSymLink* attribute is set to
9092 TRUE, the device shall also create a routing table entry with the destination address field set to the source
9093 address of the route request command frame and the next hop field set to the address of the previous device
9094 that transmitted the command frame. The status field shall be set to ACTIVE. The device shall then issue a
9095 route reply command frame to the source of the route request command frame. In the case that the device
9096 already has a route discovery table entry for the source address and route request identifier pair, the device
9097 shall determine if the path cost in the route request command frame is less than the forward cost stored in
9098 the route discovery table entry. The comparison is made by first computing the link cost from the previous
9099 device that sent this frame, as described in section 3.6.3.1, then adding it to the path cost value in the route
9100 request command frame. If this value is greater than the value in the route discovery table entry, the frame
9101 shall be dropped and no further processing is required. Otherwise, the forward cost and sender address
9102 fields in the route discovery table are updated with the new cost and the previous device address from the
9103 route request command frame.

9104 If the *nwkSymLink* attribute is set to TRUE and the received route request command frame is a unicast
9105 route request, the device shall also create a routing table entry with the destination address field set to the
9106 source address of the route request command frame and the next hop field set to the address of the previous
9107 device that transmitted the command frame. The status field shall be set to ACTIVE. The device shall then
9108 respond with a route reply command frame. In either of these cases, if the device is responding on behalf of
9109 one of its end device children, the responder address in the route reply command frame payload shall be set
9110 equal to the address of the end device child and not of the responding device.

When a device with routing capacity is not the destination of the received route request command frame, it shall determine if a route discovery table entry (see Table 3.58) exists with the same route request identifier and source address field. If no such entry exists, one shall be created. The route request timer shall be set to expire in *nwkRouteDiscoveryTime* OctetDurations. If a routing table entry corresponding to the routing address of the destination exists and its status is not ACTIVE or VALIDATION_UNDERWAY, the status shall be set to DISCOVERY_UNDERWAY. If no such entry exists and the frame is a unicast route request, an entry shall be created and its status set to DISCOVERY_UNDERWAY. If the frame is a many-to-one route request, the device shall also create a routing table entry with the destination address field equal to the source address of the route request command frame by setting the next hop field to the address of the previous device that transmitted the command frame. If the frame is a many-to-one route request (*i.e.* the many-to-one sub-field of the command options field of the command frame payload has a non-zero value), the many-to-one field in the routing table entry shall be set to TRUE, the route record required field shall be set to TRUE³, and the no route cache flag shall be set to TRUE if the many-to-one sub-field of the command options field of the command frame payload has a value of 2 or to FALSE if it has a value of 1. If the routing table entry is new, or if the no route cache flag is set to TRUE, or if the next hop field changed, the route record required field shall be set to TRUE, otherwise it remains unchanged. The status field shall be set to ACTIVE. When the route request timer expires, the device deletes the route request entry from the route discovery table. When this happens, the routing table entry corresponding to the routing address of the destination shall also be deleted, if its status field has a value of DISCOVERY_UNDERWAY and there are no other entries in the route discovery table created as a result of a route discovery for that destination address.

If an entry in the route discovery table already exists, the path cost in the route request command frame shall be compared to the forward cost value in the route discovery table entry. The comparison is made by computing the link cost from the previous device, as described in section 3.6.3.1, and adding it to the path cost value in the route request command frame. If this path cost is greater, the route request command frame is dropped and no further processing is required. Otherwise, the forward cost and sender address fields in the route discovery table are updated with the new cost and the previous device address from the route request command frame. Additionally, the path cost field in the route request command frame shall be updated with the cost computed for comparison purposes. If the *nwkSymLink* attribute is set to TRUE and the received route request command frame is a unicast route request, the device shall also update any routing table entry with the destination address field set to the source address of the route request command frame, and the next hop field set to the address of the previous device that transmitted the command frame. The status field shall be set to ACTIVE. The device shall then broadcast the route request command frame using the MCPS-DATA.request primitive.

When broadcasting a route request command frame, the NWK layer shall delay retransmission by a random jitter amount calculated using the formula:

$$2 \times R[nwkMinRREQJitter, nwkMaxRREQJitter]$$

where R is a random function on the interval. The units of this jitter amount are milliseconds. Implementers may adjust the jitter amount so that route request command frames arriving with large path cost are delayed more than frames arriving with lower path cost. The NWK layer shall retry the broadcast *nwkRREQRetries* times after the original relay resulting in a maximum of *nwkRREQRetries* + 1 relays per relay attempt. Implementers may choose to discard route request command frames awaiting retransmission in the case that a frame with the same source and route request identifier arrives with a lower path cost than the one awaiting retransmission.

The device shall also set the status field of the routing table entry corresponding to the routing address of the destination field in the payload to DISCOVERY_UNDERWAY. If no such entry exists, it shall be created.

³ CCB 1487

9158 When replying to a route request with a route reply command frame, a device that has a route discovery ta-
9159 ble entry corresponding to the source address and route request identifier of the route request shall con-
9160 struct a command frame with the frame type field set to 0x01. The source address field of the NWK header
9161 shall be set to the 16-bit network address of the current device and the destination address field shall be set
9162 to the value of the sender address field from the corresponding route discovery table entry. The device con-
9163 structing the route reply shall populate the payload fields in the following manner.

- 9164 • The NWK command identifier shall be set to route reply.
9165 • The route request identifier field shall be set to the same value found in the route request identifier
9166 field of the route request command frame.
9167 • The originator address field shall be set to the source address in the NWK header of the route request
9168 command frame.
9169 • Using the sender address field from the route discovery table entry corresponding to the source address
9170 in the NWK header of the route request command frame, the device shall compute the link cost as de-
9171 scribed in section 3.6.3.1. This link cost shall be entered in the path cost field.

9172 The route reply command frame is then unicast to the destination by using the MCPS-DATA.request primi-
9173 tive and the sender address obtained from the route discovery table as the next hop.

9174 **3.6.3.5.3 Upon Receipt of a Route Reply Command Frame**

9175 On receipt of a route reply command frame, a device shall perform the following procedure.

9176 If the receiving device has no routing capacity and its NIB attribute *nwkUseTreeRouting* has a value of
9177 TRUE, it shall send the route reply as though it were a data frame being forwarded using tree routing. If the
9178 receiving device has no routing capacity and its NIB attribute *nwkUseTreeRouting* has a value of FALSE, it
9179 shall discard the command frame. Before forwarding the route reply command frame the device shall up-
9180 date the path cost field in the payload by computing the link cost from the next hop device to itself as de-
9181 scribed in section 3.6.3.1 and adding this to the value in the route reply path cost field.

9182 To support legacy devices, a route reply received with a radius of 1 shall NOT be dropped. It shall continue
9183 to be processed as follows.

9184 If the receiving device has routing capacity, it shall check whether it is the destination of the route reply
9185 command frame by comparing the contents of the originator address field of the command frame payload
9186 with its own address. If it is, it shall search its route discovery table for an entry corresponding to the route
9187 request identifier in the route reply command frame payload. If there is no such entry, the route reply
9188 command frame shall be discarded and route reply processing shall be terminated. If a route discovery table
9189 entry exists, the device shall search its routing table for an entry with a destination address field equal to the
9190 routing address corresponding to the responder address in the route reply command frame. If there is no
9191 such routing table entry, the route reply command frame shall be discarded and, if a route discovery table
9192 entry corresponding to the route request identifier in the route reply command frame exists, it shall also be
9193 removed and route reply processing shall be terminated. If a routing table entry and a route discovery table
9194 entry exist and if the status field of the routing table entry is set to DISCOVERY_UNDERWAY, it shall be
9195 changed to VALIDATION_UNDERWAY if the routing table entry's GroupId flag is TRUE or to ACTIVE
9196 otherwise; the next hop field in the routing table shall be set to the previous device that forwarded the route
9197 reply command frame. The residual cost field in the route discovery table entry shall be set to the path cost
9198 field in the route reply payload.

9199 If the status field was already set to ACTIVE or VALIDATION_UNDERWAY, the device shall compare
9200 the path cost in the route reply command frame to the residual cost recorded in the route discovery table
9201 entry, and update the residual cost field and next hop field in the routing table entry if the cost in the route
9202 reply command frame is smaller. If the path cost in the route reply is not smaller, the route reply shall be
9203 discarded and no further processing shall take place. Note that NLDE data requests may be processed as
9204 soon as the first valid route is determined.

9205 If the device receiving the route reply is not the destination, the device shall find the route discovery table
9206 entry corresponding to the originator address and route request identifier in the route reply command frame
9207 payload. If no such route discovery table entry exists, the route reply command frame shall be discarded. If
9208 a route discovery table entry exists, the path cost value in the route reply command frame and the residual
9209 cost field in the route discovery table entry shall be compared. If the route discovery table entry value is
9210 less than the route reply value, the route reply command frame shall be discarded.

9211 Otherwise, the device shall find the routing table entry with a destination address field equal to the routing
9212 address corresponding to the responder address in the route reply command frame. In this case, it is an error
9213 if the route discovery table entry exists and there is no corresponding routing table entry, and the route re-
9214 sponse command frame should be discarded. The routing table entry shall be updated by replacing the next hop
9215 field with the address of the previous device that forwarded the route reply command frame. The route dis-
9216 covery table entry shall also be updated by replacing the residual cost field with the value in the route reply
9217 command frame.

9218 Whenever the receipt of a route reply causes the next hop field of the corresponding routing table entry to
9219 be modified, and the routing table entry's GroupId flag is TRUE, the device shall set the expiration time
9220 field of the corresponding route discovery table entry to expire in *nwkcWaitBeforeValidation* OctetDurations
9221 if the device is the destination of the route reply and *nwkcRouteDiscoveryTime* OctetDurations if it is
9222 not.

9223 After updating its own route entry, the device shall forward the route reply to the destination. Before for-
9224 warding the route reply, the path cost value shall be updated. The sender shall find the next hop to the route
9225 reply's destination by searching its route discovery table for the entry matching the route request identifier
9226 and the source address and extracting the sender address. It shall use this next hop address to compute the
9227 link cost as described in section 3.6.3.1. This cost shall be added to the path cost field in the route reply.
9228 The destination address in the command frame NWK header shall be set to the next hop address and the
9229 frame shall be unicast to the next hop device using the MCPS-DATA.request primitive. The DstAddr pa-
9230 rameter of the MCPS-DATA.request primitive shall be set to the next-hop address from the route discovery
9231 table.

9232 If the value of the *nwkSymLink* attribute of the NIB has a value of TRUE, the NWK layer shall, upon re-
9233 laying the route reply command frame, also create a reverse routing table entry if such an entry does not yet
9234 exist. The value of the destination address field of the routing table entry shall correspond to the value of
9235 the originator address field of the route reply command frame. The status field shall have a value of AC-
9236 TIVE. The next-hop address field shall have a value corresponding to the next hop address in the route re-
9237 sponse command being relayed, as determined in the previous paragraph. If the reverse routing table entry al-
9238 ready exists the next-hop address field shall be updated, if necessary.

9239 **3.6.3.5.4 Initiation and Processing of a Route Record Command Frame**

9240 If the NWK layer of a ZigBee router or ZigBee coordinator is initiating a unicast data frame as a result of
9241 an NLDE-DATA.request from the next higher layer and the many-to-one field of the routing table entry
9242 corresponding to the destination address of the frame has a value of TRUE, then the NWK layer shall ex-
9243 amine the route record required field of that same routing table entry. If the route record required field also
9244 has a value of TRUE, the NWK shall unicast a route record command to the destination before transmitting
9245 the data frame.

9246 If the NWK layer of a ZigBee router or ZigBee coordinator is forwarding a unicast data frame on behalf of
9247 one of its end device children and the many-to-one field of the destination's routing table entry has a value
9248 of TRUE, then the device shall unicast a route record command to the destination before relaying the data
9249 frame.

9250 An optional optimization is possible in which the router or coordinator may keep track of which of its end
9251 device children have received source routed data frames from a particular concentrator device and can
9252 thereby reduce the number of route record commands it transmits to that concentrator on behalf of its end
9253 device children.

9254 Each relay node that receives the route record command shall append its network address to the command
9255 payload, increment the relay count, and forward the message. If no next hop is available, or if delivery to
9256 the next hop fails, or if there is insufficient space in the payload for the network address, the command
9257 frame shall be discarded and no error command shall be generated.

9258 Upon receipt of the route record command by the destination, the route shall be stored in the source route
9259 table. Any existing source routes to the message source or intermediary nodes shall be replaced by the new
9260 route information.

9261 **3.6.3.6 Upon Expiration of a Route Discovery Table Entry**

9262 When a route discovery table entry is created, the expiration timer shall be set to expire in *nwkRouteDiscoveryTime* OctetDurations. For entries whose GroupId flag in the corresponding entry in the routing table is TRUE, when a route reply is received that causes the next hop to change, the expiration time field of the corresponding route discovery table entry is set to expire in *nwkWaitBeforeValidation* OctetDurations if the device is the destination of the route reply and *nwkRouteDiscoveryTime* OctetDurations if it is not.
9263 When the timer expires, the device shall delete the entry from the route discovery table. If the device is the
9264 originator of the route request and the routing table entry corresponding to the destination address has a
9265 Status field value of VALIDATION_UNDERWAY, then the device shall transmit a message to validate
9266 the route: either the message-buffered pending route discovery or a network status command with a status
9267 code of 0x0a (validate route). If the routing table entry corresponding to the destination address has any
9268 Status field value other than ACTIVE or VALIDATION_UNDERWAY and there are no other entries in
9269 the route discovery table corresponding to that routing table entry, the routing table entry shall also be de-
9270 letted.

9275 **3.6.3.7 Route Maintenance**

9276 A device NWK layer shall maintain a failure counter for each neighbor to which it has an outgoing link,
9277 i.e., to which it has been required to send data frames. If the outgoing link is classified as a failed link, then
9278 the device shall respond as described in the following paragraphs. Implementers may choose a simple fail-
9279 ure-counting scheme to generate this failure counter value or they may use a more accurate time-windowed
9280 scheme. Note that it is important not to initiate repair too frequently since repair operations may flood the
9281 network and cause other traffic disruptions. The procedure for retiring links and ceasing to keep track of
9282 their failure counter is out of the scope of this specification.

9283 **3.6.3.7.1 In Case of Link Failure**

9284 If a failed link is encountered while a device is forwarding a unicast data frame using a routing table entry
9285 with the many-to-one field set to TRUE, a network status command frame with status code of 0x0c indi-
9286 cating many-to-one route failure shall be generated. The destination address field in the NWK header of the
9287 network status command frame shall be equal to the destination address field in the NWK header of the
9288 frame causing the error. The destination address field of the network status command payload shall be
9289 equal to the source address field in the NWK header of the frame causing the error. The network status
9290 command frame shall be unicast to a random router neighbor using the MCPS-DATA.request primitive.
9291 Because it is a many-to-one route, all neighbors are expected to have a routing table entry to the destina-
9292 tion. Upon receipt of the network status command frame, if no routing table entry for the destination is
9293 present, or if delivery of the network status command frame to the next hop in the routing table entry fails,
9294 the network status command frame shall again be unicast to a random router neighbor using the
9295 MCPS-DATA.request primitive. The radius counter in the NWK header will limit the maximum number of
9296 times the network status command frame is relayed. Upon receipt of the network status command frame by
9297 its destination it shall be passed up to the next higher layer using the NLME-NWK-STATUS.indication
9298 primitive. Many-to-one routes are not automatically rediscovered by the NWK layer due to route errors.

9299 If a failed link is encountered while the device is forwarding a unicast frame using normal unicast routing,
9300 the device shall issue a network status command frame back to the source device of the frame with a status
9301 code indicating the reason for the failure (see Table 3.42), and issue an NLME-NWK-STATUS.indication
9302 to the next higher layer with a status code indicating the reason for the failure.

9303 On receipt of a network status command frame by a router that is the intended destination of the command where the status code field of the command frame payload has a value of 0x01 or 0x02 indicating a link failure, the NWK layer will remove the routing table entry corresponding to the value of the destination address field of the command frame payload, if one exists, and inform the next higher layer of the failure using the NLME-NWK-STATUS.indication using the same status code.
9304
9305
9306
9307
9308 On receipt of a network status command frame by a router that is the parent of an end device that is the intended destination, where the status code field of the command frame payload has a value of 0x01 or 0x02 indicating a link failure, the NWK layer will remove the routing table entry corresponding to the value of the destination address field of the command frame payload, if one exists. It will then relay the frame as usual to the end device.
9309
9310
9311
9312
9313 On receipt of a network status command frame by an end device, the NWK layer shall inform the next higher layer of the failure using the NLME-NWK-STATUS.indication.
9314
9315 If an end device encounters a failed link to its parent, the end device shall inform the next higher layer using the NLME-NWK-STATUS.indication primitive with a Status parameter value of 0x09 indicating parent link failure (see Table 3.42). Similarly if a ZigBee router without routing capacity for which *nwkUseTreeRouting* has a value of TRUE encounters a failed link to its parent, it shall inform the next higher layer using the NLME-NWK-STATUS.indication primitive with a Status parameter value of 0x09 indicating parent link failure.
9316
9317
9318
9319
9320

3.6.4 Scheduling Beacon Transmissions

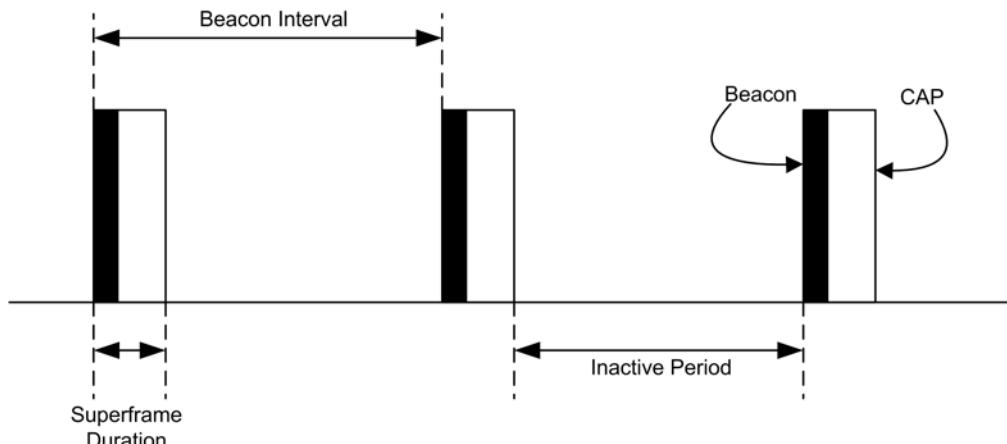
9321 Beacon scheduling is necessary in a multi-hop topology to prevent the beacon frames of one device from colliding with either the beacon frames or the data transmissions of its neighboring devices. Beacon scheduling is necessary when implementing a tree topology but not a mesh topology, as beaconing is not permitted in ZigBee mesh networks.

3.6.4.1 Scheduling Method

9322 The ZigBee coordinator shall determine the beacon order and superframe order for every device in the network (see [B1] for more information on these attributes). Because one purpose of multi-hop beaconing networks is to allow routing nodes the opportunity to sleep in order to conserve power, the beacon order shall be set much larger than the superframe order. Setting the attributes in this manner makes it possible to schedule the active portion of the superframes of every device in any neighborhood such that they are non-overlapping in time. In other words, time is divided into approximately (*macBeaconInterval/macSuperframeDuration*) non-overlapping time slots, and the active portion of the superframe of every device in the network shall occupy one of these non-overlapping time slots. An example of the resulting frame structure for a single beaconing device is shown in Figure 3.48.

9323
9324
9325
9326
9327
9328
9329
9330
9331
9332
9333
9334
9335
9336
9337

Figure 3.48 Typical Frame Structure for a Beaconing Device



9338

9339 The beacon frame of a device shall be transmitted at the start of its non-overlapping time slot, and the
9340 transmit time shall be measured relative to the beacon transmit time of the parent device. This time offset
9341 shall be included in the beacon payload of every device in a multi-hop beaconing network (see section 3.6.7
9342 for a complete list of beacon payload parameters). Therefore a device receiving a beacon frame shall know
9343 the beacon transmission time of both the neighboring device and the parent of the neighboring device, since
9344 the transmission time of the parent may be calculated by subtracting the time offset from the timestamp of
9345 the beacon frame. The receiving device shall store both the local timestamp of the beacon frame and the
9346 offset included in the beacon payload in its neighbor table. The purpose of having a device know when the
9347 parent of its neighbor is active is to maintain the integrity of the parent-child communication link by alleviating
9348 the hidden node problem. In other words, a device will never transmit at the same time as the parent
9349 of its neighbor.

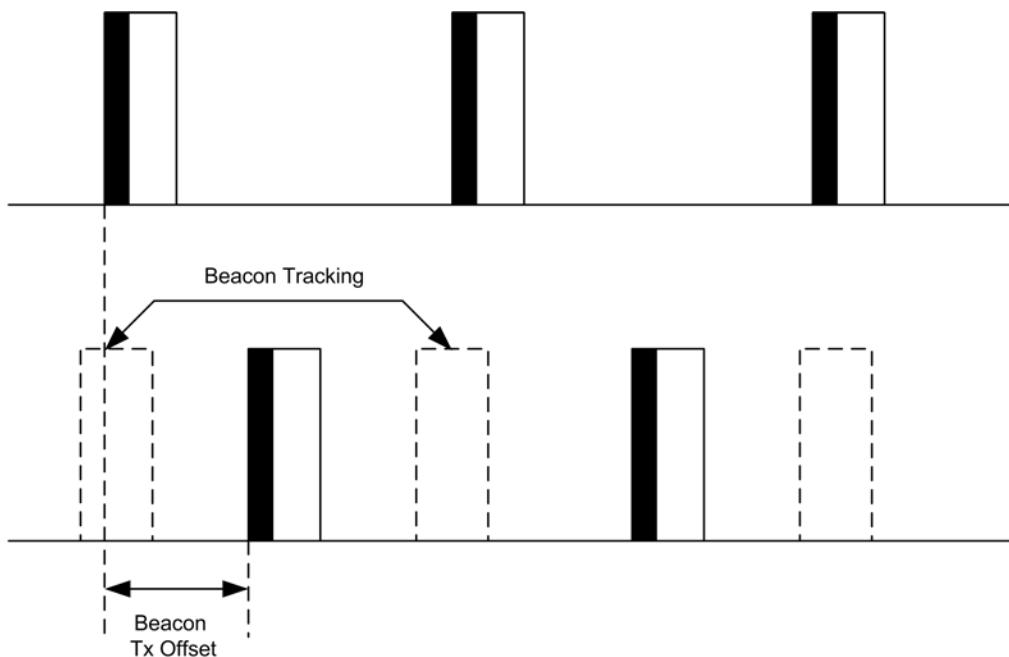
9350 Communication in a tree network shall be accomplished using the parent-child links to route along the tree.
9351 Since every child tracks the beacon of its parent, transmissions from a parent to its child shall be completed
9352 using the indirect transmission technique. Transmissions from a child to its parent shall be completed during
9353 the CAP of the parent. Details for the communication procedures can be found in IEEE 802.15.4-2003
9354 [B1].

9355 A new device wishing to join the network shall follow the procedure outlined in section 3.6.1.4. In the process
9356 of joining the network, the new device shall build its neighbor table based on the information collected
9357 during the MAC scan procedure. Using this information, the new device shall choose an appropriate time
9358 for its beacon transmission and CAP (the active portion of its superframe structure) such that the active
9359 portion of its superframe structure does not overlap with that of any neighbor or of the parent of any
9360 neighbor. If there is no available non-overlapping time slot in the neighborhood, the device shall not trans-
9361 mit beacons and shall operate on the network as an end device. If a non-overlapping time slot is available,
9362 the time offset between the beacon frames of the parent and the new device shall be chosen and included in
9363 the beacon payload of the new device. Any algorithm for selecting the beacon transmission time that avoids
9364 beacon transmission during the active portion of the superframes of its neighbors and their parents may be
9365 employed, as interoperability will be ensured.

9366 To counteract drift, the new device shall track the beacon of its parent and adjust its own beacon transmis-
9367 sion time such that the time offset between the two remains constant. Therefore, the beacon frames of every
9368 device in the network are essentially synchronized with those of the ZigBee coordinator. Figure 3.49 illus-
9369 trates the relationship between the active superframe portions of a parent and its child.

9370

Figure 3.49 Parent-Child Superframe Positioning Relationship



9371

9372 The density of devices that can be supported in the network is inversely proportional to the ratio of the superframe order to the beacon order. The smaller the ratio, the longer the inactive period of each device and
9373 the more devices that can transmit beacon frames in the same neighborhood. It is recommended that a tree
9374 network utilize a superframe order of 0, which, when operating in the 2.4 GHz band, gives a superframe
9375 duration of 15.36 ms and a beacon order of between 6 and 10, which, in the 2.4 GHz band, gives a beacon
9376 interval between 0.98304s and 15.72864s. Using these superframe and beacon order values, a typical duty
9377 cycle for devices in the network will be between ~2% and ~0.1% regardless of the frequency band.
9378

9379

3.6.5 Broadcast Communication

9380 This section specifies how a broadcast transmission is accomplished within a ZigBee network. Any device
9381 within a network may initiate a broadcast transmission intended for a number of other devices that are part
9382 of the same network. A broadcast transmission is initiated by the local APS sub-layer entity through the use
9383 of the NLDE-DATA.request primitive by setting the DstAddr parameter to a broadcast address as shown in
9384 Table 3.59, or by the NWK layer through the use of these same broadcast addresses in the construction of
9385 an outgoing NWK header. (Note that broadcast transmission for link status and route request command
9386 frames is handled differently as described in section 3.6.3.4 and section 3.6.3.5.2 respectively.)

9387

Table 3.59 Broadcast Addresses

Broadcast Address	Destination Group
0xffff	All devices in PAN
0xfffe	Reserved
0xffffd	<i>macRxOnWhenIdle</i> = TRUE
0xffffc	All routers and coordinator

Broadcast Address	Destination Group
0xffffb	Low power routers only
0xffff8 - 0xffffa	Reserved

9388

9389 To transmit a broadcast MSDU, the NWK layer of a ZigBee router or ZigBee coordinator issues an
9390 MCPS-DATA.request primitive to the MAC sub-layer with the DstAddrMode parameter set to 0x02
9391 (16-bit network address) and the DstAddr parameter set to 0xffff. For a ZigBee end device, the MAC des-
9392 tination address of the broadcast frame shall be set equal to the 16-bit network address of the parent of the
9393 end device. The PANId parameter shall be set to the PANId of the ZigBee network. This specification does
9394 not support broadcasting across multiple networks. Broadcast transmissions shall not use the MAC
9395 sub-layer acknowledgement; instead, a passive acknowledgement mechanism may be used. Passive
9396 acknowledgement means that every ZigBee router and ZigBee coordinator keeps track of which of its
9397 neighboring devices have successfully relayed the broadcast transmission. The MAC sub-layer acknowl-
9398 edgement is disabled by setting the acknowledged transmission flag of the TxOptions parameter to FALSE.
9399 All other flags of the TxOptions parameter shall be set based on the network configuration.

9400 The ZigBee coordinator, each ZigBee router and those ZigBee end devices with *macRxOnWhenIdle* equal
9401 to TRUE, shall keep a record of any new broadcast transaction that is either initiated locally or received
9402 from a neighboring device. This record is called the broadcast transaction record (BTR) and shall contain at
9403 least the sequence number and the source address of the broadcast frame. The broadcast transaction records
9404 are stored in the *nwkBroadcastTransactionTable* (BTT) as shown in Table 3.60.

9405

Table 3.60 Broadcast Transaction Record

Field Name	Size	Description
Source Address	2 bytes	The 16-bit network address of the broadcast initiator.
Sequence Number	1 byte	The NWK layer sequence number of the initiator's broadcast.
Expiration Time	1 byte	A countdown timer indicating the number of seconds until this entry expires; the initial value is <i>nwkNetworkBroadcastDeliveryTime</i> .

9406 When a device receives a broadcast frame from a neighboring device, it shall compare the destination address of the frame with its device type. If the destination address does not correspond to the device type of the receiver as outlined in Table 3.59, the frame shall be discarded. If the destination address corresponds to the device type of the receiver, the device shall compare the sequence number and the source address of the broadcast frame with the records in its BTT.
 9407
 9408
 9409
 9410

9411 If the device has a BTR of this particular broadcast frame in its BTT, it may update the BTR to mark the neighboring device as having relayed the broadcast frame. It shall then drop the frame. If no record is
 9412 found, it shall create a new BTR in its BTT and may mark the neighboring device as having relayed the
 9413 broadcast. The NWK layer shall then indicate to the higher layer that a new broadcast frame has been received using the NLDE-DATA.indication. If the device is a ZigBee router (ZR) or a ZigBee Coordinator
 9414 (ZC) and the radius field is greater than zero; then the frame shall be retransmitted. Otherwise it shall be
 9415 dropped. Before the retransmission, it shall wait for a random time period called broadcast jitter. This time
 9416 period shall be bounded by the value of the *nwkMaxBroadcastJitter* attribute. ZigBee end devices with
 9417 *macRxOnWhenIdle* equal to FALSE shall not participate in the relaying of broadcast frames and need not
 9418 maintain a BTT for broadcast frames that they originate.
 9419

9420
 9421 If, on receipt of a broadcast frame, the NWK layer finds that the BTT is full and contains no expired entries, then the frame should be dropped. In this situation the frame should not be retransmitted, nor should it
 9422 be passed up to the next higher layer.
 9423

9424 A ZigBee coordinator or ZigBee router operating in a non-beacon-enabled ZigBee network shall retransmit
 9425 a previously broadcast frame at most *nwkMaxBroadcastRetries* times. If the device does not support passive
 9426 acknowledgement, then it shall retransmit the frame exactly *nwkMaxBroadcastRetries* times. If the
 9427 device supports passive acknowledgement and any of its neighboring devices have not relayed the broad-
 9428 cast frame within *nwkPassiveAckTimeout* OctetDurations then it shall continue to retransmit the frame up
 9429 to a maximum of *nwkMaxBroadcastRetries* times.

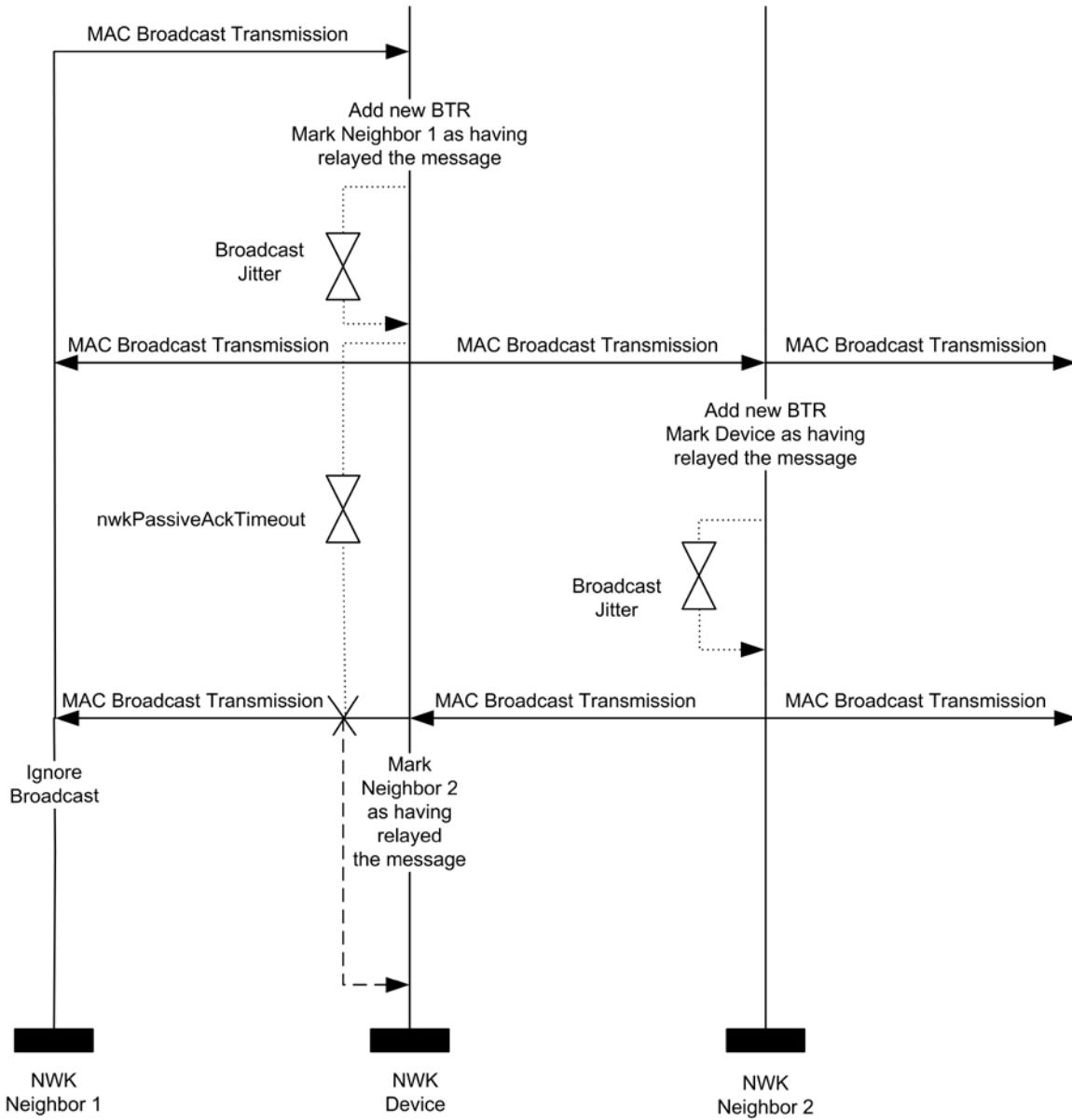
9430 A device should change the status of a BTT entry after *nwkNetworkBroadcastDeliveryTime* OctetDurations
 9431 have elapsed since its creation. The entry status should change to expired and thus the entry can be over-
 9432 written if required when a new broadcast is received.

9433 When a ZigBee router that has the *macRxOnWhenIdle* MAC PIB attribute set to FALSE receives a broad-
 9434 cast transmission, it shall use a different procedure for retransmission than the one outlined above. It shall
 9435 retransmit the frame without delay to each of its neighbors individually, using a MAC layer unicast, that is,
 9436 with the DstAddr parameter of the MCPS-DATA.request primitive set to the address of each neighbor
 9437 device and not to the broadcast address. Similarly, a router or coordinator with the *macRxOnWhenIdle* MAC
 9438 PIB attribute set to TRUE, which has one or more neighbors with the *macRxOnWhenIdle* MAC PIB attrib-
 9439 ute set to FALSE, shall, in the case where the destination address is 0xffff denoting broadcast to all devices,
 9440 retransmit the broadcast frame to each of these neighbors in turn as a MAC layer unicast in addition to per-
 9441 forming the more general broadcast procedure spelled out in the previous paragraphs. Indirect transmission,
 9442 as described in IEEE 802.15.4-2003 [B1], may be employed to ensure that these unicasts reach their desti-
 9443 nation.

9444 Every ZigBee router shall have the ability to buffer at least 1 frame at the NWK layer in order to facilitate
9445 retransmission of broadcasts.

9446 Figure 3.50 shows a broadcast transaction between a device and two neighboring devices.

9447 **Figure 3.50 Broadcast Transaction Message Sequence Chart**



9448
9449

3.6.6 Multicast Communication

9451 This section specifies how multicast transmission is accomplished within a ZigBee network. Multicast address
9452 ing is accomplished using 16-bit multicast group IDs. A multicast group is a collection of nodes, all
9453 registered under the same multicast group ID, that are physically separated by a hop distance of no more
9454 than a given radius, known as the MaxNonMemberRadius. A multicast message is sent to a particular desti-
9455 nation group and is received by all members of that group. Only data frames are multicast — no NWK
9456 command frames are multicast.

9457 Multicast frames are propagated through the network by both members and non-members of the destination
9458 multicast group. A packet may be sent in one of two modes as indicated by a mode flag in the packet which
9459 determines the method of relay to the next hop. If the original message was created by a member of the
9460 group, it is considered to be in ‘Member Mode’ and is relayed by means of broadcasts. If the original mes-
9461 sage was created by a non-member of the group, it is considered to be in ‘Non-Member Mode’ and is re-
9462 layed by means of unicasts towards a group member. Once a non-member message reaches any member of
9463 the destination group, it is instantly transformed into a Member Mode type relay for the duration of the life
9464 of the packet regardless of who relays it next.

9465 Multicast messages may be originated by end devices but are not sent to devices where *macRxOnWhenIdle*
9466 is equal to FALSE.

9467 **3.6.6.1 The Group ID Table**

9468 The NWK layer of a device may maintain a group ID table, *nwkGroupIDTable*, accessible as an attribute of
9469 the NIB as shown in Table 3.49. If the *nwkGroupIDTable* NIB attribute is present then it shall contain a set
9470 of 16-bit group identifiers for groups of which the device is a member.

9471 Note that the optional *nwkGroupIDTable* NIB attribute has a functional overlap with the mandatory APS
9472 group table (see Table 2-18). If a device maintains both tables, and thereby expects to use NWK-layer mul-
9473 ticast as a method for receiving group-addressed frames, it must assure that each 16-bit group identifiers
9474 that appears in the APS group table also appears in the NWK group table.

9475 Note also that from an implementation perspective, it would be wasteful to duplicate the list of group iden-
9476 tifiers across layers and it is assumed that implementers will find a way to combine the APS and NWK
9477 group tables to avoid waste.

9478 **3.6.6.2 Upon Receipt of a Multicast Frame from the Next Higher 9479 Layer**

9480 If an NLDE-DATA.request is received by the NWK layer from its next higher layer and the multicast con-
9481 trol field is 0x01, the NWK layer shall determine whether an entry exists in the *nwkGroupIDTable* having a
9482 group identifier field matching the destination address of the frame. If a matching entry is found, the NWK
9483 layer shall multicast the frame according to the procedure outlined in section 3.6.6.2.1. If a matching entry
9484 is not found, the frame shall be initiated as a non-member mode multicast using the procedure outlined in
9485 section 3.6.6.2.2.

9486 **3.6.6.2.1 Initiating a Member Mode Multicast**

9487 The NWK layer shall set the multicast mode sub-field of the multicast control field to 0x01 (member
9488 mode). If the BTT table is full and contains no expired entries, the message shall not be sent and the NLDE
9489 shall issue the NLDE-DATA.confirm primitive with a status value of BT_TABLE_FULL. If the BTT is not
9490 full or contains an expired BTR, a new BTR shall be created with the local node as the source and the mul-
9491 ticast frame’s sequence number. The message shall then be transmitted according to the procedure de-
9492 scribed in the final paragraph of section 3.6.6.3.

9493 **3.6.6.2.2 Initiating a Non-Member Mode Multicast**

9494 The NWK layer shall set the multicast mode sub-field of the multicast control field to 0x00 (non-member
9495 mode). Then, the NWK layer shall check its routing table for an entry corresponding to the GroupID desti-
9496 nation of the frame. If there is such an entry, the NWK layer shall examine the entry's status field. If the
9497 status is ACTIVE, then the device shall (re)transmit the frame. If the status is VALIDA-
9498 TION_UNDERWAY, then the status shall be changed to ACTIVE, the device shall transmit the frame acc-
9499 cording to the procedure described in the final paragraph of section 3.6.6.4, and the NLDE shall issue the
9500 NLDE-DATA.confirm primitive with the status value received from the MCPS-DATA.confirm primitive.
9501 If there is no routing table entry corresponding to the GroupID destination of the frame and the value of the
9502 DiscoverRoute parameter is 0x00 (suppress route discovery), the frame shall be discarded and the NLDE
9503 shall issue the NLDE-DATA.confirm primitive with a status value of ROUTE_DISCOVERY_FAILED. If
9504 the DiscoverRoute parameter has a value of 0x01 (enable route discovery) and there is no routing table en-
9505 try corresponding to the GroupID destination of the frame, then the device shall initiate route discovery
9506 immediately as described in section 3.6.3.5.1. The frame may optionally be buffered pending route discov-
9507 ery. If it is not buffered, the frame shall be discarded and the NLDE shall issue the NLDE-DATA.confirm
9508 primitive with a status value of FRAME_NOT_BUFFERED.

9509 **3.6.6.3 Upon Receipt of a Member Mode Multicast Frame**

9510 When a device receives a member mode multicast frame from a neighboring device, it shall compare the
9511 sequence number and the source address of the multicast frame with the records in its BTT. If the device
9512 has a BTR of this particular multicast frame in its BTT it shall discard the frame. If no record is found and
9513 the BTT is full and contains no expired entries, it shall discard the frame. If no record is found and the BTT
9514 is not full or contains an expired BTR, it shall create a new BTR and continue processing the message as
9515 outlined in the following paragraph.

9516 When a member mode multicast frame has been received from a neighbor and added to the BTT, the NWK
9517 layer shall then determine whether an entry exists in the *nwkGroupIDTable* whose group identifier field
9518 matches the destination group ID of the frame. If a matching entry is found, the message shall be passed to
9519 the next higher layer, the multicast mode sub-field of the multicast control field shall be set to 0x01 (mem-
9520 ber mode), the value of the NonmemberRadius sub-field shall be set to the value of the MaxNonmember-
9521 Radius sub-field in the multicast control field, and the message shall be transmitted as outlined in the fol-
9522 lowing paragraph.

9523 If a matching entry is not found, the NWK layer shall examine the frame's multicast NonmemberRadius
9524 field. If the value of the NonmemberRadius sub-field of the multicast field is 0 the message shall be dis-
9525 carded, along with the newly added BTR. Otherwise, the NonmemberRadius sub-field shall be decremen-
9526 ted if it is less than 0x07 and the frame shall be transmitted as outlined in following paragraphs. If, as a re-
9527 sult of being decremented, this value falls to 0, the frame shall not, under any circumstances, be retransmit-
9528 ted.

9529 Each member mode multicast message shall be transmitted *nwkMaxBroadcastRetries* times. For member
9530 mode multicast frames that did not originate on the local device, the initial transmission shall be delayed by
9531 a random time bounded by the value of the *nwkMaxBroadcastJitter* attribute. A device shall delay a period
9532 of *nwkPassiveAckTimeout* OctetDurations between retransmissions of a particular member mode multicast
9533 message. Unlike broadcasts, there is no passive acknowledgement for multicasts. ZigBee end devices shall
9534 not participate in the relaying of multicast frames.

9535 To transmit a member mode multicast MSDU, the NWK layer issues an MCPS-DATA.request primitive to
9536 the MAC sub-layer with the DstAddrMode parameter set to 0x02 (16-bit network address) and the DstAddr
9537 parameter set to 0xffff, which is the broadcast network address. The PANId parameter shall be set to the
9538 PANId of the ZigBee network. Member mode multicast transmissions shall not use the MAC sub-layer
9539 acknowledgement or the passive acknowledgement used for broadcasts. The MAC sub-layer acknowl-
9540 edgement is disabled by setting the acknowledged transmission flag of the TxOptions parameter to FALSE.
9541 All other flags of the TxOptions parameter shall be set based on the network configuration.

3.6.6.4 Upon Receipt of a Non-Member Mode Multicast Frame

When a device receives a non-member mode multicast frame from a neighboring device, the NWK layer shall determine whether an entry exists in the *nwkGroupIDTable* having a group identifier field that matches the destination address of the frame. If a matching entry is found, the multicast control field shall be set to 0x01 (member mode) and the message shall be processed as if it had been received as a member mode multicast. If no matching *nwkGroupIDTable* entry is found, the device shall check its routing table for an entry corresponding to the GroupID destination of the frame. If there is no such routing table entry, the message shall be discarded. If there is such an entry, the NWK layer shall examine the entry's status field. If the status is ACTIVE, the device shall (re)transmit the frame. If the status is VALIDATION_UNDERWAY, the status shall be changed to ACTIVE and the device shall (re)transmit the frame. To transmit a non-member mode multicast MSDU, the NWK layer issues an MCPS-DATA.request primitive to the MAC sublayer with the DstAddrMode parameter set to 0x02 (16-bit network address) and the DstAddr parameter set to the next hop as determined from the matching routing table entry. The PANId parameter shall be set to the PANId of the ZigBee network. The MAC sub-layer acknowledgement shall be enabled by setting the acknowledged transmission flag of the TxOptions parameter to TRUE. All other flags of the TxOptions parameter shall be set based on the network configuration.

3.6.7 NWK Information in the MAC Beacons

This section specifies how the NWK layer uses the beacon payload of a MAC sub-layer beacon frame to convey NWK layer-specific information to neighboring devices.

The beacon payload shall contain the information shown in Table 3.61. This enables the NWK layer to provide additional information to new devices that are performing network discovery and allows these new devices to more efficiently select a network and a particular neighbor to join. Refer to section 3.6.1.4.1.1 for a detailed description of the network discovery procedure.

Table 3.61 NWK Layer Information Fields

Name	Type	Valid Range	Description
Protocol ID	Integer	0x00 – 0xff	This field identifies the network layer protocols in use and, for purposes of this specification, shall always be set to 0, indicating the ZigBee protocols. The value 0xff shall also be reserved for future use by the ZigBee Alliance.
Stack profile	Integer	0x00 – 0x0f	A ZigBee stack profile identifier.
<i>nwkcProtocolVersion</i>	Integer	0x00 – 0x0f	The version of the ZigBee protocol.
Router capacity	Boolean	TRUE or FALSE	This value is set to TRUE if this device is capable of accepting join requests from router-capable devices and is set to FALSE otherwise.

Name	Type	Valid Range	Description
Device depth	Integer	0x00 – 0x0f	The network depth of this device. A value of 0x00 indicates that this device is the ZigBee coordinator for the network.
End device capacity	Boolean	TRUE or FALSE	This value is set to TRUE if the device is capable of accepting join requests from end devices seeking to join the network and is set to FALSE otherwise.
<i>nwkExtendedPANId</i>	64-bit extended address	0x0000000000000001 – 0xfffffffffffffe	The globally unique ID for the PAN of which the beaconing device is a member. By default, this is the 64-bit IEEE address of the ZigBee coordinator that formed the network, but other values are possible and there is no required structure to the address.
TxOffset	Integer	0x000000 – 0xffff	This value indicates the difference in time, measured in symbols, between the beacon transmission time of the device and the beacon transmission time of its parent; This offset may be subtracted from the beacon transmission time of the device to calculate the beacon transmission time of the parent. This parameter is set to the default value of 0xFFFF in beaconless networks.
<i>nwkUpdateId</i>	Integer	0x00 - 0xFF	This field reflects the value of <i>nwkUpdateId</i> from the NIB.

9566

9567 The NWK layer of the ZigBee coordinator shall update the beacon payload immediately following network formation. All other ZigBee devices shall update it immediately after the join is completed and any time the network configuration (any of the parameters specified in Table 3.10) changes. The beacon payload is written 9568 into the MAC sub-layer PIB using the MLME-SET.request primitive. The *macBeaconPayloadLength* attribute 9569 is set to the length of the beacon payload, and the octet sequence representing the beacon payload is 9570 written into the *macBeaconPayload* attribute. The formatting of the bit sequence representing the beacon 9571 payload is shown in Figure 3.50. 9572 9573

9574

Figure 3.51 Format of the MAC Sub-Layer Beacon Payload

Bits: 0–7	8–11	12–15	16–17	18	19–22	23	24–87	88–111	112–119
Protocol ID	Stack profile	<i>nwk cProtocol Version</i>	Re-served	Router capacity	Device depth	End device capacity	<i>nwk Extended PANId</i>	Tx Offset	<i>Nwk UpdateId</i>

9575

3.6.8 Persistent Data

9576 Devices operating in the field may be restarted either manually or programmatically by maintenance personnel, or may be restarted accidentally for any number of reasons, including localized or network-wide power failures, battery replacement during the course of normal maintenance, impact, and so on. The following information should be preserved across resets in order to maintain an operating network:

- 9580 • The device's PAN Id and Extended PAN Id.
- 9581 • The device's 16-bit network address.
- 9582 • If *nwkAddrAlloc* is equal to 0, a device shall save the following information for each associated router child in the neighbor table:
 - 9583 • The 64-bit IEEE address
 - 9584 • 16-bit network address
- 9585 • For each device in the *nwkNeighborTable* of the NIB with a device type set to 0x02 (ZigBee End Device), the following shall be saved:
 - 9586 • The 64-bit IEEE address
 - 9587 • 16-bit network address
 - 9588 • The End Device Configuration value
 - 9589 • Device Timeout value
- 9590 • If the device is an end device, the *nwkParentInformation* value in the NIB.
- 9591 • For end devices, the 16-bit network address of the parent device.
- 9592 • The stack profile in use.
- 9593 • The device depth.

9594 The method by which these data are made to persist is beyond the scope of this specification.

9597

3.6.9 Low Power Routers (LPR)

9598 Low power routers are defined as routers operating on batteries for multiple years by regularly powering off their radios. LPRs shall be recognized by high power routers (HPR) looking at the following capability information bit-fields (see Table 3.52) during the joining phase:

- 9601 • Device type set to 1
- 9602 • Receive on when idle set to FALSE

9603 LPR devices should be able to receive network command frames that are broadcast in the network. This can be achieved by setting the destination address in the NWK header to the broadcast address for all routers and coordinators (see Table 3.59).

9606

3.6.10 End Device Aging and Management

9607
9608
9609
9610
9611

The end device and router relationship is established via MAC association or NWK rejoin, and can be dissolved via a leave command. However there are a number of ways in which the relationship can get broken, where router parent and end device do not agree. For example the router parent may think it is still the router parent for an end device when in fact the end device has switched to a new parent, or the router parent may age out the child since it has had no communication with it for an extended period of time.

9612
9613
9614

Router parents have a finite amount of local resources to store end device information. As such it is desirable to clean out old entries to allow for new end devices to join. End devices shall be aged out by the router according to the rules defined below.

9615

3.6.10.1 End Device Aging Mechanism

9616
9617
9618

A router parent must age neighbor table entries for end devices. It is important to note that prior versions of this specification did not have this requirement and thus legacy devices exist that do not have this child aging mechanism.

9619
9620
9621
9622

A router parent shall keep track of the amount of real time that has passed and decrement the Timeout counter value for each end device entry in its neighbor table until the value reaches 0. When a neighbor table entry's Timeout counter value reaches 0, the router parent shall delete the entry from the neighbor table.

9623
9624

End Devices may periodically send a keepalive message to reset the Timeout counter value. See section 3.6.10.3 for details.

9625

3.6.10.2 Establishing the Timeout

9626
9627
9628

A router shall initially set the timeout for all end devices according to the default value of *nwkEndDeviceTimeoutDefault* in Table 3.49. The following describes how an end device may update this value from the default.

9629
9630
9631

After joining or rejoining the network the end device shall send an End Device Timeout Request command to its parent. This shall be done even if the end device is joining or rejoining to the same parent. The message shall include their timeout period and configuration.

9632

Routers shall process the End Device Timeout Request command as follows.

9633
9634
9635

1. If the Requested Timeout Enumeration value in the frame is not within the valid range, it shall generate an End Device Timeout Response command with a status of INCORRECT_VALUE and no further processing of the message shall take place.

9636
9637
9638

2. The parent shall find the neighbor table entry for the sending device and verify that the entry corresponds to an end device. If no entry is found or the entry is not an end device, then the message shall be dropped and no further processing should take place.

9639
9640
9641
9642
9643

3. The received value shall be converted into an actual timeout amount. This shall be done by obtaining the actual timeout value for the corresponding Requested Timeout Enumeration in Table 3.44. The value shall be converted from minutes into seconds if it is not already a value in seconds. The parent shall set the Timeout Counter and Device Timeout values of the neighbor table entry to the converted value.

9644
9645
9646

4. The parent shall set the End Device Configuration information in the neighbor table for the corresponding end device's entry to the value of the End Device Configuration field in the received message.

9647
9648
9649

5. The parent shall generate an End Device Timeout Response command with a status of SUCCESS. It shall fill in the value of the *Parent Information Bitmask* field according to the keepalive methods it supports.

9650

An End Device that receives an End Device Timeout Response Command shall process it as follows.

- 9651 1. If the status is SUCCESS it shall set the *nwkParentInformation* value in the NIB to value of the
9652 Parent Information field of the received command. No further processing shall take place.
9653 2. If the End Device receives the command with a status value other than SUCCESS, it shall assume
9654 its timeout value has not been configured on the parent.

9655 End Devices may receive no End Device Timeout Response command at all if they are communicating
9656 with a legacy device that does not have support for this command. They shall treat this the same as re-
9657 ceiving an End Device Timeout Response with a non-SUCCESS status code.

9658 3.6.10.3 End Device Keepalive

9659 All end devices (including RxOnWhenIdle=TRUE) that have received an End Device Timeout Response
9660 Command with a status of SUCCESS may periodically send a keepalive to their router parent to insure they
9661 remain in the router's neighbor table.

9662 The keepalive message will refresh the timeout on the parent device so that the parent does not delete the
9663 child from its neighbor table. The period for sending the keepalive to the router parent shall be deter-
9664 mined by the manufacturer of the device and is not specified by this standard. It is recommended that the
9665 period allows the end device to send 3 keepalive messages during the Device Timeout period. This will
9666 help insure that a single missed keepalive message will not age out the end device on the router parent.

9667 There are two keepalive mechanisms described below. The method the end device uses depends on the
9668 support of the router parent. The router parent will indicate its support in the End Device Timeout Re-
9669 sponse command frame and this information will be stored in the NIB.

9670 When an End Device needs to send a keepalive message, it shall examine the *nwkParentInformation* value
9671 in the NIB. If bit 0 has a value of 1 (indicating support of the MAC data poll keepalive) then the device
9672 shall send a MAC data poll command unicast to its parent.

9673 Otherwise if the value of bit 1 has a value of 1, then the device shall send an End Device Timeout Request
9674 command as a unicast to refresh the keepalive timer. If the transmission is successful, the device shall
9675 wait for macResponseWaitTime for an End Device Timeout Response from its parent. If the transmission
9676 was unsuccessful, or if no End Device Timeout Response command is received, or if the status field indi-
9677 cates a value other than SUCCESS, the end device shall generate a NLME-NWK-STATUS.indication with
9678 a code of 0x09 (Parent Link Failure).

9679 3.6.10.4 MAC Data Poll Processing

9680 A router whose *nwkParentInformation* in the NIB has bit 1 set to 0, shall support the MAC Data poll as an
9681 End Device keepalive. A router is not required to support this method. If it does not it must support the
9682 End Device Timeout Request method.

9683 Upon receipt of an MLME-POLL.Indication the router parent shall examine its neighbor table and do **one**
9684 of the following:

- 9685 1. If there is no entry in the neighbor table corresponding to the DeviceAddress of the
9686 MLME-Poll.Indication primitive, then the device shall construct a leave message. The destina-
9687 tion NWK address shall be set to the value of the MAC source of the MAC data poll. See section
9688 3.6.10.4.1 for more information on the leave message. The message shall be added to the indi-
9689 rect transaction queue of the MAC layer.
- 9690 2. If there is an entry in the neighbor table for the sending device's MAC source, then the local de-
9691 vice shall set the Timeout counter value to the value of the *End Device Keepalive Timeout* value,
9692 and it shall set the Keepalive Received value to TRUE.

9693 When an End Device sends a MAC Data poll command it shall assume that the parent has knowledge of
9694 the end device and the Timeout Counter associated with the end device has been reset in the parent's
9695 neighbor table. The End Device will behave per reference [B1] with regard to the data pending bit in the
9696 MAC Ack, and will follow standard processing of any leave message that may be received after sending a
9697 data poll.

9698 **3.6.10.4.1 Sending a Leave Message**

9699 A router shall send a leave message when it wants to inform an end device it is no longer a parent to the
9700 end device. The leave message shall be one of the following messages:

9701 1. NWK Leave Request

- 9702 a. A device that chooses to send a NWK leave request shall set fields of the NWK Com-
9703 mand as follows.
- 9704 i. The destination IEEE address sub-field of the frame control shall be set to 0, in-
9705 dicating that no destination IEEE address is present.
 - 9706 ii. The destination IEEE address field shall not be present in the message.
 - 9707 iii. The request sub-field of the command options field shall be set to 1.
 - 9708 iv. The rejoin request sub-field of the command shall be set to 1.

9709 2. ZDO Mgmt_Leave_Req

- 9710 a. A device that chooses to send a ZDO Mgmt_Leave_Req shall set the fields of the ZDO Mgmt_leave_req command as follows:
- 9711 i. The Device Address field shall be set to NULL (0x0000000000000000)
 - 9712 ii. The Remove Children Bit shall be set to 0.
 - 9713 iii. The Rejoin bit shall be set to 1.
- 9714 b. The Acknowledgement request sub-field of the APS Frame control field shall be set to 0
9715 (no acknowledgement requested).

9717 **3.6.10.5 Setting the End Device Timeout on the Router Parent**

9719 A router shall set the default values for Timeout Counter and End Device Keepalive Timeout to the
9720 time-span indicated by *nwkEndDeviceTimeoutDefault* as converted to seconds.

9721 After successfully joining or rejoining the network and receiving the network key, an End Device shall
9722 send an End Device Timeout Request command to its router parent indicating its desired timeout. Upon
9723 receipt and successful processing of the End Device Timeout Request router parents shall update the
9724 timeout values accordingly. See section 3.6.10.2 for details.

9725 Legacy devices will not send an End Device Timeout Request and thus will receive the default timeout.

9726 **3.6.10.6 Local End Device Timeout**

9727 An end device may keep track of its timeout using the following mechanism:

- 9728 1. The end device shall find the corresponding neighbor table entry for its router parent.
- 9729 2. It shall decrement the Timeout Counter value in the Neighbor Table entry based on the amount of
9730 real time that has passed, until that value reaches 0.
- 9731 3. If the Timeout Counter reaches a value of 0, it shall assume that its parent has timed out the de-
9732 vice.

9733 If the end device has determined that it has been timed out, it can choose to perform a rejoin to get back on
9734 the network as described in section 3.6.1.4.2. Alternatively it is permissive for an end device to always
9735 perform a rejoin without keep tracking of its local end device timeout.

9736 There is no requirement that the end device re-establish connectivity with the network if it has determined
9737 that it has reached the timeout value established with its router parent. An end device may choose to de-
9738 lay rejoining the network until it is appropriate, for example when the end device has data it needs to send.

3.6.10.7 Persistent Values on the Parent Router

The router parent is expected to persistently store the end device information in the neighbor table (see section 3.6.8).

3.6.10.8 Reboot and Child Aging

On reboot routers shall set the Timeout Counter value for each end device in its neighbor table to the entry's value of Device Timeout. In other words, end devices shall be given a full time period for aging out.

On reboot it is recommended end devices immediately initiate a keep-alive message to verify connectivity to their parent.

3.6.10.9 Diagrams Illustrating End Device Management

Figure 3.52 Initial Setup of the End Device Timeout

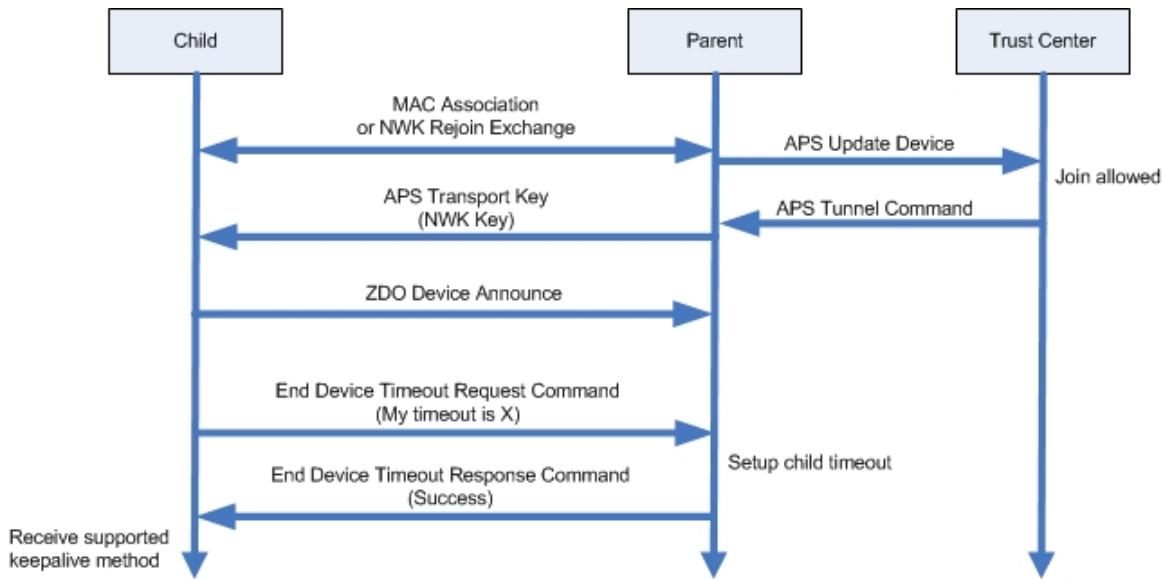
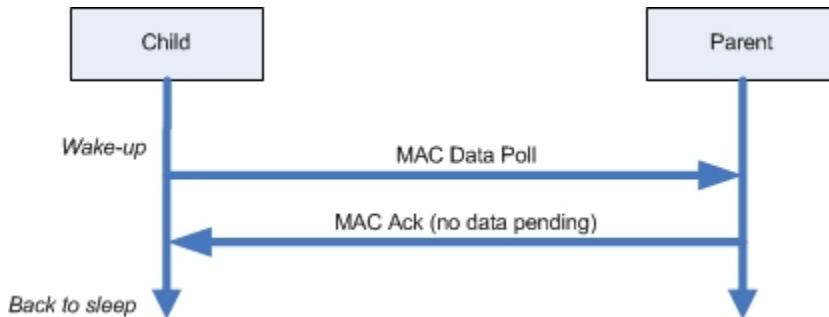


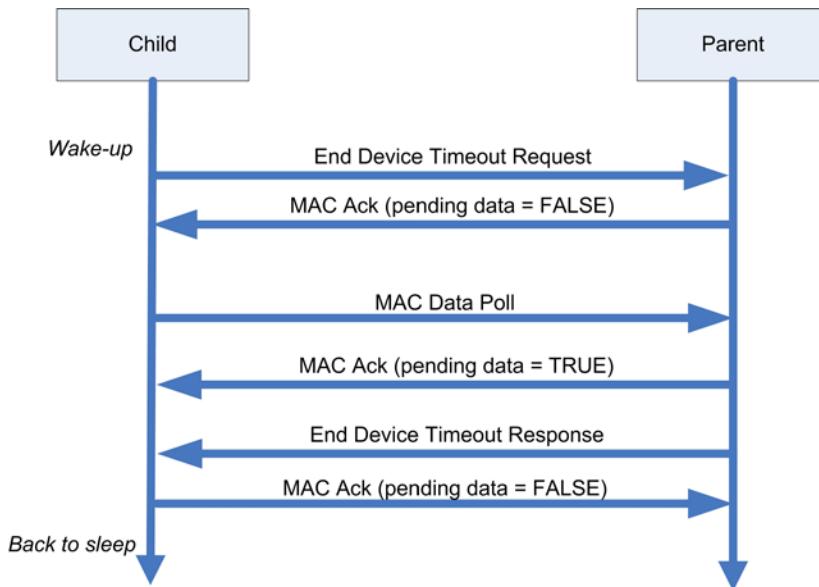
Figure 3.52 shows an end device joining into a network and the series of message exchanges. After the end device has joined and has a copy of the NWK key, it will send a NWK command of End Device Request to the parent and check for a response.

Figure 3.53 Child Keepalive: MAC Data Poll Method



9758 Figure 3.53 shows normal operation of a child talking to a parent that supports the MAC Data Poll
9759 Keepalive Method. When the data pending bit is unset in the MAC acknowledgement, the end device can
9760 assume that the parent still remembers the device.

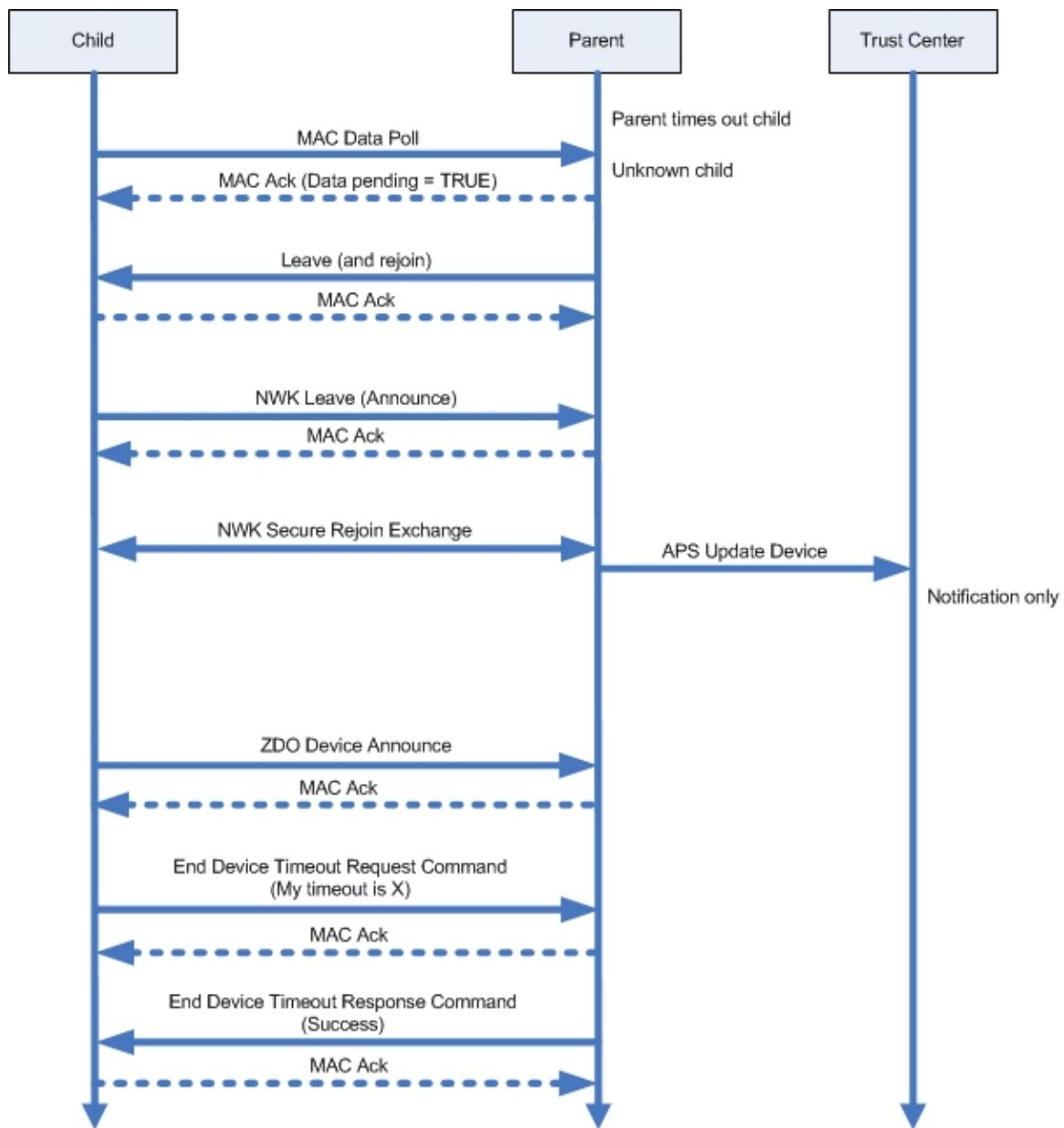
9761 **Figure 3.54 Child Keepalive: End Device Timeout Request Method**



9762
9763 Figure 3.54 shows normal operation of a child talking to a parent that supports the End Device Timeout
9764 Request keepalive method. T.

9765

Figure 3.55 Aging out Children: MAC Data Poll Method - Secure Rejoin

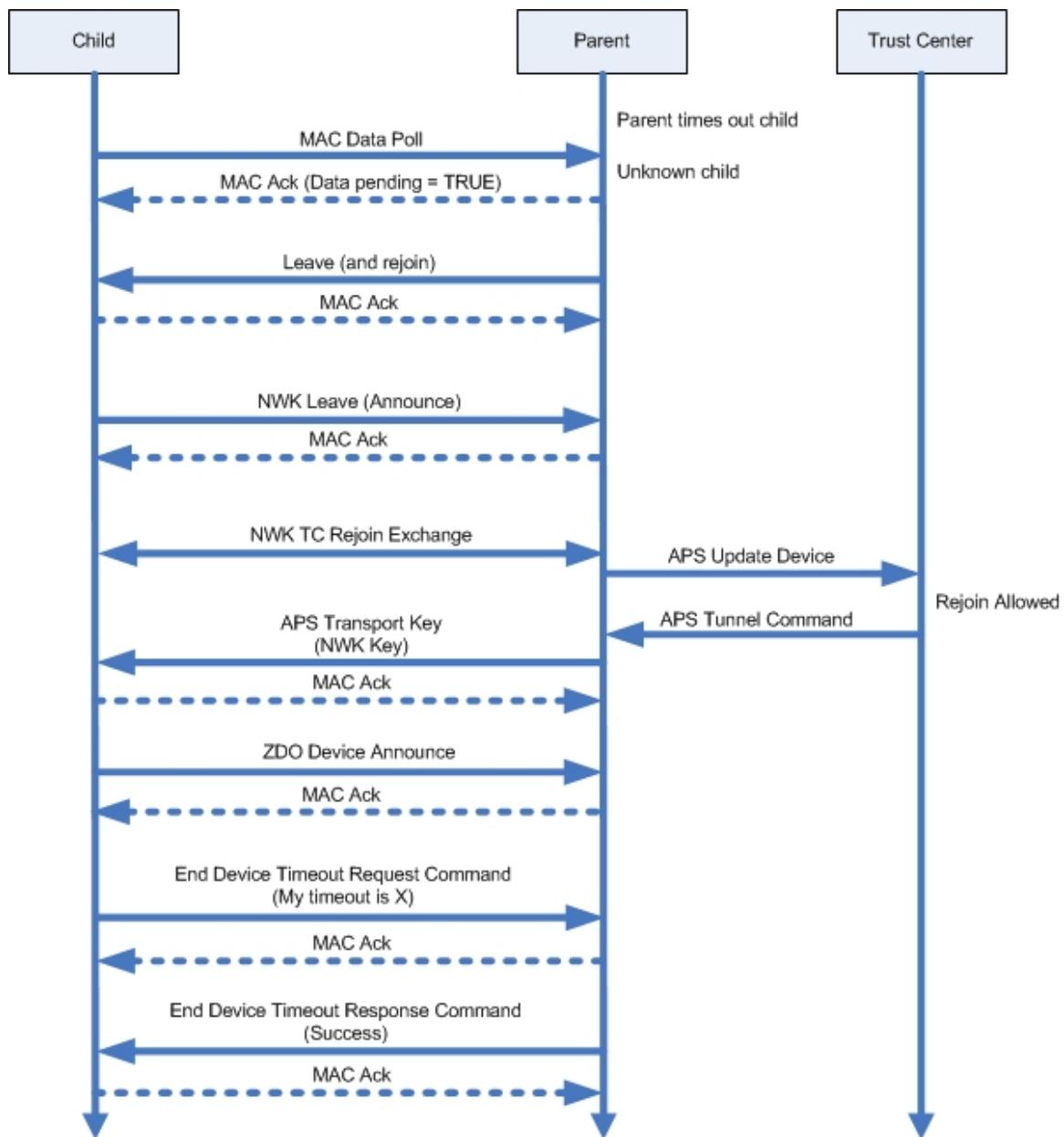


9766

9767

9768

Figure 3.56 Aging out Children: MAC Data Poll - Trust Center Rejoin

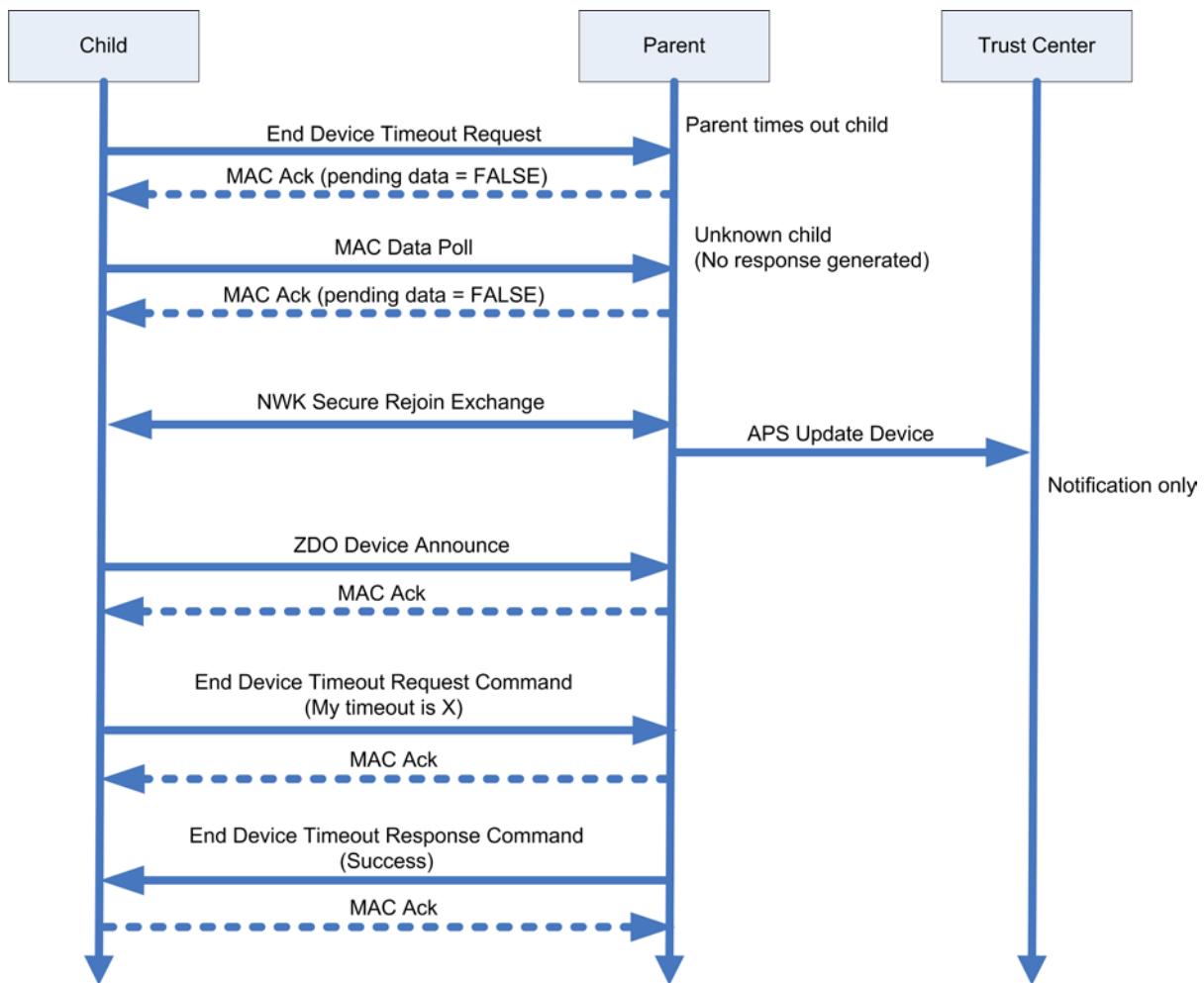


9769
9770

9771 Figure 3.55 and Figure 3.56 show what happens when a parent that supports the MAC data poll keepalive
9772 method, ages out the child. The parent will indicate to the child that it has a pending message for the child by
9773 setting the data pending bit to TRUE in the MAC acknowledgement. The parent will then transmit a leave
9774 message to the device with the rejoin bit set to TRUE. The device will announce leaving the network and
9775 perform a rejoin. Figure 3.55 shows a secure rejoin while Figure 3.56 shows a Trust Center Rejoin. After
9776 the rejoin is successful the device will send the NWK Command End Device Timeout Request and receive a
9777 response.

9778

Figure 3.57 Aging out Children: End Device Timeout Request Method - Secure Rejoin

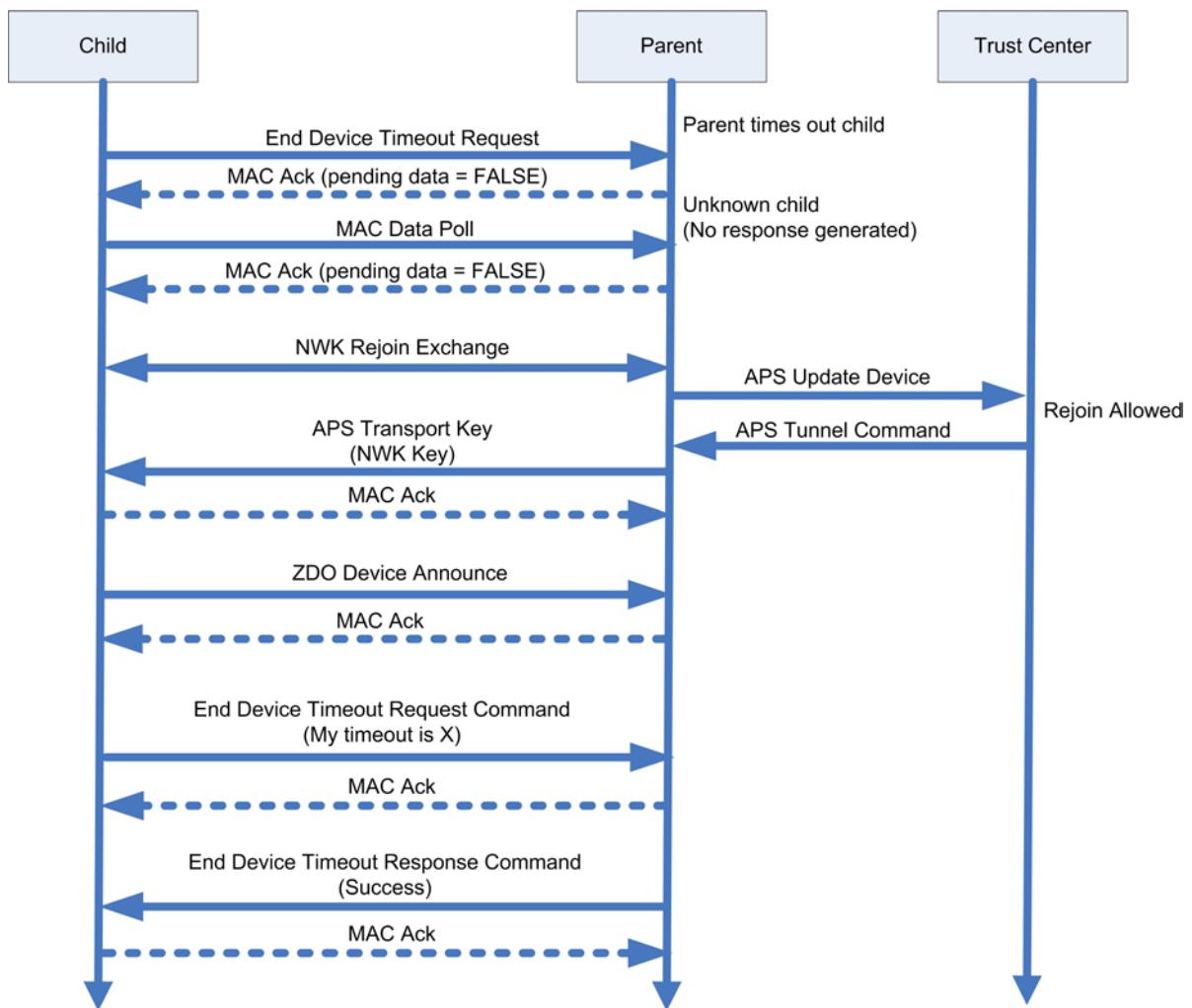


9779

9780

9781

Figure 3.58 Aging out Children: End Device Timeout Request Method - Trust Center Rejoin



9783 Figure 3.57 and Figure 3.58 shows what happens when an end device is aged out of the parent's table with
9784 a parent that supports the End Device Timeout Request method. An end device sends an End Device
9785 Timeout Request and receives no response. Afterwards it will perform a rejoin. Figure 3.57 shows a
9786 secure rejoin while Figure 3.58 shows a Trust Center rejoin. Once the device has completed the rejoin it
9787 will send a NWK command End Device timeout request and receive the response.

9788

3.6.10.10 Trust Center Rejoin or Secure Rejoin

9789 An end device that has detected it has been aged out of its parent's child table may choose to use either a
9790 Secure Rejoin or a Trust Center rejoin. The choice to use one or the other is up to the implementation but
9791 can be based on whether it may have missed a network key update. A device that has missed a network
9792 key update will have to use a Trust Center Rejoin. However in a case where that situation has not oc-
9793 curred, a Secure Rejoin will complete more quickly and can be used instead. It is possible that an end de-
9794 vice may try both methods to insure it can get back on the network.

9795

3.7 NWK Layer Status Values

9796 Network (NWK) layer confirmation primitives often include a parameter that reports on the status of the
9797 request to which the confirmation applies. Values for NWK layer Status parameters appear in Table 3.62.

Table 3.62 NWK Layer Status Values

Name	Value	Description
SUCCESS	0x00	A request has been executed successfully.
INVALID_PARAMETER	0xc1	An invalid or out-of-range parameter has been passed to a primitive from the next higher layer.
INVALID_REQUEST	0xc2	The next higher layer has issued a request that is invalid or cannot be executed given the current state of the NWK layer.
NOT_PERMITTED	0xc3	An NLME-JOIN.request has been disallowed.
STARTUP_FAILURE	0xc4	An NLME-NETWORK-FORMATION.request has failed to start a network.
ALREADY_PRESENT	0xc5	A device with the address supplied to the NLME-DIRECT-JOIN.request is already present in the neighbor table of the device on which the NLME-DIRECT-JOIN.request was issued.
SYNC_FAILURE	0xc6	Used to indicate that an NLME-SYNC.request has failed at the MAC layer.
NEIGHBOR_TABLE_FULL	0xc7	An NLME-JOIN-DIRECTLY.request has failed because there is no more room in the neighbor table.
UNKNOWN_DEVICE	0xc8	An NLME-LEAVE.request has failed because the device addressed in the parameter list is not in the neighbor table of the issuing device.
UNSUPPORTED_ATTRIBUTE	0xc9	An NLME-GET.request or NLME-SET.request has been issued with an unknown attribute identifier.
NO_NETWORKS	0xca	An NLME-JOIN.request has been issued in an environment where no networks are detectable.
Reserved	0xcb	
MAX_FRM_COUNTER	0xcc	Security processing has been attempted on an outgoing frame, and has failed because the frame counter has reached its maximum value.
NO_KEY	0xcd	Security processing has been attempted on an outgoing frame, and has failed because no key was available with which to process it.

Name	Value	Description
BAD_CCM_OUTPUT	0xce	Security processing has been attempted on an outgoing frame, and has failed because the security engine produced erroneous output.
Reserved	0xcf	
ROUTE_DISCOVERY_FAILED	0xd0	An attempt to discover a route has failed due to a reason other than a lack of routing capacity.
ROUTE_ERROR	0xd1	An NLDE-DATA.request has failed due to a routing failure on the sending device or an NLME-ROUTE-DISCOVERY.request has failed due to the cause cited in the accompanying NetworkStatusCode.
BT_TABLE_FULL	0xd2	An attempt to send a broadcast frame or member mode multicast has failed due to the fact that there is no room in the BTT.
FRAME_NOT_BUFFERED	0xd3	An NLDE-DATA.request has failed due to insufficient buffering available. A non-member mode multicast frame was discarded pending route discovery.

9799

9800
9801
9802
9803
9804
9805
9806
9807
9808
9809
9810
9811
9812
9813
9814

This page intentionally left blank.

CHAPTER 4 SECURITY SERVICES SPECIFICATION

4.1 Document Organization

The remaining portions of this document specify in greater detail the various security services available within the ZigBee stack. Basic definitions and references are given in clause 4.2. A general description of the security services is given in section 4.2.1. In this clause, the overall security architecture is discussed; basic security services provided by each layer of this architecture are introduced. Sections 4.2.2 and 4.2.3 give the ZigBee Alliance's security specifications for the Network (NWK) layer and the Application Support Sublayer (APS) layer, respectively. These clauses introduce the security mechanisms, give the primitives, and define any frame formats used for security purposes. Section 4.5 describes security elements common to the NWK and APS layers. Section 4.6 provides a basic functional description of the available security features. Finally, annexes provide technical details and test vectors needed to implement and test the cryptographic mechanisms and protocols used by the NWK and APS layers.

4.2 General Description

Security services provided for ZigBee include methods for key establishment, key transport, frame protection, and device management. These services form the building blocks for implementing security policies within a ZigBee device. Specifications for the security services and a functional description of how these services shall be used are given in this document.

4.2.1 Security Architecture and Design

In this clause, the security architecture is described. Where applicable, this architecture complements the security services that are already present in the IEEE Std. 802.15.4 802 [B1] security specification.

4.2.1.1 Security Assumptions

The level of security provided by the ZigBee security architecture depends on the safekeeping of the symmetric keys, on the protection mechanisms employed, and on the proper implementation of the cryptographic mechanisms and associated security policies involved. Trust in the security architecture ultimately reduces to trust in the secure initialization and installation of keying material and to trust in the secure processing and storage of keying material.

9842 Implementations of security protocols, such as key establishment, are assumed to properly execute the
9843 complete protocol and not to leave out any steps thereof. Random number generators are assumed to operate
9844 as expected. Furthermore, it is assumed that secret keys do not become available outside the device in
9845 an unsecured way. That is, a device will not intentionally or inadvertently transmit its keying material to
9846 other devices unless the keying material is protected, such as during key-transport. During initial key
9847 transport the keying material used for protection may be a well-known key, thus resulting in a brief moment
9848 of vulnerability where the key could be obtained by any device. Alternatively, the initial key
9849 transport may be done using a pre-shared secret key that is passed out-of-band from the ZigBee network.
9850 The following caveat in these assumptions applies: due to the low-cost nature of *ad hoc* network devices,
9851 one cannot generally assume the availability of tamper-resistant hardware. Hence, physical access to a de-
9852 vice may yield access to secret keying material and other privileged information, as well as access to the
9853 security software and hardware.

9854 Due to cost constraints, ZigBee has to assume that different applications using the same radio are not logi-
9855 cally separated (for example, by using a firewall). In addition, from the perspective of a given device it is
9856 not even possible (barring certification) to verify whether cryptographic separation between different ap-
9857 plications on another device — or even between different layers of the communication stack thereof — is
9858 indeed properly implemented. Hence, one must assume that separate applications using the same radio trust
9859 each other; that is, there is no cryptographic task separation. Additionally, lower layers (for example, APS,
9860 NWK, or MAC) are fully accessible by any of the applications. These assumptions lead to an open trust
9861 model for a device; different layers of the communication stack and all applications running on a single de-
9862 vice trust each other.

9863 In summary:

- 9864 • The provided security services cryptographically protect the interfaces between different devices
9865 only.
9866 • Separation of the interfaces between different stack layers on the same device is arranged
9867 non-cryptographically, via proper design of security service access points.

9868 4.2.1.2 Security Design Choices

9869 The open trust model (as described in section 4.2.1.1) on a device has far-reaching consequences. It allows
9870 re-use of the same keying material among different layers on the same device and it allows end-to-end se-
9871 curity to be realized on a device-to-device basis rather than between pairs of particular layers (or even pairs
9872 of applications) on two communicating devices.

9873 However, one must also take into consideration whether one is concerned with the ability of malevolent
9874 network devices to use the network to transport frames across the network without permission.

9875 These observations lead to the following architectural design choices:

9876 First, the principle that “*the layer that originates a frame is responsible for initially securing it*” is estab-
9877 lished. For example, if a NWK command frame needs protection, NWK layer security shall be used.

9878 Second, if protection from theft of service is required (*i.e.*, from malevolent network devices), NWK layer
9879 security shall be used for all frames, except those passed between a router and a newly joined device (until
9880 the newly joined device receives the active network key). Thus, only a device that has joined the network
9881 and successfully received the active network key will be able to have its messages communicated more
9882 than one hop across the network.

9883 Third, due to the open trust model, security can be based on the reuse of keys by each layer. For example,
9884 the active network key shall be used to secure APS layer broadcast frames or NWK layer frames. Reuse of
9885 keys helps reduce storage costs.

9886 Fourth, end-to-end security is provided such that it is possible for only source and destination devices to
9887 access messages protected by a shared key. This ensures that routing of messages between the two devices
9888 with the shared key can be independent of trust considerations.

9889 Fifth, to simplify interoperability of devices, the base security level used by all devices in a given network,
9890 and by all layers of a device, shall be the same. If an application needs more security for its payload than is
9891 provided by network level security, it can establish application level security with another device. There

9892 are several policy decisions which any real implementation must address correctly. Application profiles
9893 should include policies to:

9894 Handle error conditions arising from securing and unsecuring packets. Some error conditions may indicate
9895 loss of synchronization of security material, or may indicate ongoing attacks.

9896 Detect and handle loss of counter synchronization and counter overflow.

9897 Detect and handle loss of key synchronization.

9898 Expire and periodically update keys, if desired.

9899 The other security design choice is done by the device that forms a network. This device sets the security
9900 policies and processes followed by the network and devices that join the network.

9902 **4.2.1.3 Security Keys**

9903 Security amongst a network of ZigBee devices is based on “link” keys and a “network” key. Unicast com-
9904 munication between APL peer entities is secured by means of a 128-bit link key shared by two devices,
9905 while broadcast communications and any network layer communications are secured by means of a 128-bit
9906 network key shared amongst all devices in the network. The intended recipient is always aware of the exact
9907 security arrangement; that is, the recipient knows whether a frame is protected with a link key or a network
9908 key.

9909 A device shall acquire link keys either via key-transport, or pre-installation (for example, during factory in-
9910 stallation). A device shall acquire a network key via key-transport. Some application profiles have also de-
9911 veloped out of band mechanisms or key negotiation protocols used for generating link keys or network keys
9912 on devices. Ultimately, security between devices depends on secure initialization and installation of these
9913 keys.

9914 There is one type of network key; however, it can be used in either distributed or centralized security mod-
9915 els. The security model controls how a network key is distributed; and may control how network frame
9916 counters are initialized. The security model does not affect how messages are secured.

9917 There are two different types of trust center link keys: global and unique. The type of trust center link key
9918 in use by the local device shall determine how the device handles various trust center messages (APS
9919 commands), including whether to apply APS encryption. A Trust Center link key may also be used to
9920 secure APS data messages between the Trust Center and the corresponding peer device. The choice of
9921 whether to use APS security on those APS data messages is up to the higher layer application.

9922 A link key between two devices, neither of which is the trust center, is known as an application link key.

9923 The default value for the centralized security global trust center link key shall have a value of 5A 69 67 42
9924 65 65 41 6C 6C 69 61 6E 63 65 30 39 (ZigBeeAlliance09).

9925 The different types of keys used are described in Table 4.1.

9927 **Table 4.1 Link Keys Used in ZigBee Networks**

Key Name	Description
Centralized security global trust center link key	Link key used for joining centralized security networks
Distributed security global link key	Link key used for joining distributed security networks
Install code link key	Link key derived from install code from joining device to create unique trust center link key for joining

Application link key	Link key used between two devices for application layer encryption
Device Specific trust center link key	Link key used between the trust center and a device in the network. Used for trust center commands and application layer encryption.

9928

9929 In a secured network there are a variety of security services available. Prudence dictates that one would
9930 prefer to avoid re-use of keys across different security services, which otherwise could cause security leaks
9931 due to unwanted interactions. As such, these different services use a key derived from a one-way function
9932 using the link key (as specified in section 4.5.3). The use of uncorrelated keys ensures logical separation of
9933 the execution of different security protocols. The key-load key is used to protect transported link keys; the
9934 key-transport key is used to protect transported network keys. The active network key may be used by the
9935 NWK and APL layers of ZigBee. As such, the same network key and associated outgoing and incoming
9936 frame counters shall be available to all of these layers. The link keys may be used only by the APS sublayer.
9937 As such, the link key shall be available only to the APL layer.

9938 An installation code is a short code that uses an algorithm to derive the 128-bit AES key. The mechanism
9939 for deriving a key from an installation code are out of scope of this specification.

9940 **4.2.1.4 ZigBee Security Architecture**

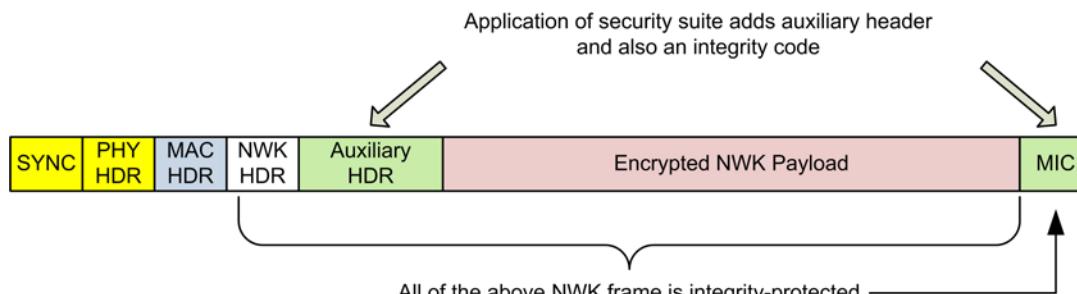
9941 The ZigBee security architecture includes security mechanisms at two layers of the protocol stack. The
9942 NWK and APS layers are responsible for the secure transport of their respective frames. Furthermore, the
9943 APS sublayer provides services for the establishment and maintenance of security relationships. The
9944 ZigBee Device Object (ZDO) manages the security policies and the security configuration of a device. Figure
9945 1.1 shows a complete view of the ZigBee protocol stack. The security mechanisms provided by the
9946 APS and NWK layers are described in this version of the specification.

9947 **4.2.2 NWK Layer Security**

9948 When a frame originating at the NWK layer needs to be secured ZigBee shall use the frame-protection
9949 mechanism given in section 4.3.1 of this specification, unless the SecurityEnable parameter of the
9950 NLDE-DATA.request primitive is FALSE, explicitly prohibiting security. For example, no NWK layer se-
9951 curity is used during transport of the NWK Key over the last hop to a joining device since APS security
9952 will be used to protect the frame. The NWK layer's frame-protection mechanism shall make use of the
9953 Advanced Encryption Standard (AES) [B8] and use CCM* as specified in Annex A. The security level ap-
9954 plied to a NWK frame shall be determined by the *nwkSecurityLevel* attribute in the NIB. Upper layers
9955 manage NWK layer security by setting up active and alternate network keys and by determining which se-
9956 curity level to use.

9957 Figure 4.1 shows an example of the security fields that may be included in a NWK frame.

9958 **Figure 4.1 ZigBee Frame with Security on the NWK Level**



9959

9960

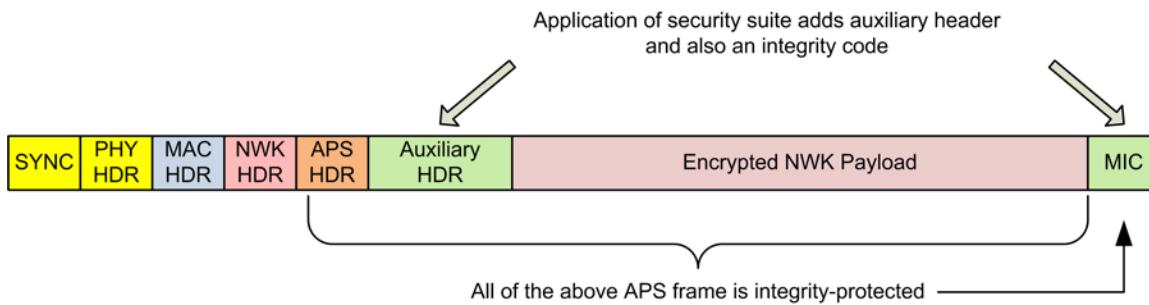
4.2.3 APL Layer Security

9961
9962
9963
9964
9965

When a frame originating at the APL layer needs to be secured, the APS sublayer shall handle security. The APS layer's frame-protection mechanism is given in section 4.4.1 of this specification. The APS layer allows frame security to be based on link keys or the network key. Figure 4.2 shows an example of the security fields that may be included in an APL frame. The APS layer is also responsible for providing applications and the ZDO with key establishment, key transport, and device management services.

9966

Figure 4.2 ZigBee Frame with Security on the APS Level



9967

4.2.3.1 Transport Key

9969
9970
9971

The transport-key service provides secured means to transport a key to another device or other devices. The secured transport-key command provides a means to transport link, or network key from a key source (for example, the Trust Center) to other devices.

9972

4.2.3.2 Update Device

9973
9974
9975
9976

The update device service provides a secure means for a router device to inform the Trust Center that a third device has had a change of status that must be updated (for example, the device joined or left the network). This is the mechanism by which the Trust Center maintains an accurate list of active network devices.

9977

4.2.3.3 Remove Device

9978
9979
9980
9981

The remove device service provides a secure means by which a Trust Center informs a router device that one of the router's children or the router itself must be removed from the network. For example, the remove device service may be employed to remove from a network a device that has not satisfied the Trust Center's security requirements for network devices.

9982

4.2.3.4 Request Key

9983
9984

The request-key service provides a secure means for a device to request an end-to-end application link key or trust center link key, from the Trust Center.

9985

4.2.3.5 Switch Key

9986
9987

The switch-key service provides a secure means for a Trust Center to inform another device that it should switch to a different active network key.

9988

4.2.3.6 Verify-Key

9989
9990

The verify-key service provides a secure means for a device to verify that the device and the Trust Center agree on the current value of the device's link key.

9991 **4.2.3.7 Confirm Key**

9992 The confirm-key service provides a secure means for a Trust Center to confirm a previous request to verify
9993 a link key.

9994

9995 **4.2.4 Trust Center Role**

9996 For security purposes, ZigBee defines the role of “Trust Center”. The Trust Center is the device trusted by
9997 devices within a network to distribute keys for the purpose of network and potentially end-to-end applica-
9998 tion configuration management. All members of the network shall recognize exactly one active Trust Cen-
9999 ter, and there shall be exactly one Trust Center in each centralized security network. The Trust Center is
10000 responsible for establishing, maintaining and updating security policies for the network.

10001 In a distributed security network, all routers have the capability to act as the Trust Center and distribute
10002 keys for network security. This distributed trust center role is used for network key distribution but not
10003 trust center link key distribution since there is not a singular trust center in the network.

10004 In some applications a device can be pre-loaded with the Trust Center address and initial Trust Center link
10005 key, or the joining device’s Trust Center link key can be installed out of band.

10006 In applications that can tolerate a moment of vulnerability, the network key can be sent via APS secured
10007 key transport using a well-known link key.

10008 In a centralized security model, the Trust Center established policies for joining devices and network secu-
10009 rity. It may require devices to be known before providing the network key update for joining, or may re-
10010 quire a preconfigured link key be installed out of band. These Trust Center policies are described in sec-
10011 tion 4.7.1.

10012 In a centralized security network a device securely communicates with its Trust Center using the current
10013 Trust Center link key.

10014 For purposes of trust management, a device only accepts a Trust Center link key or active network key
10015 originating from its Trust Center via key transport. For purposes of network management in a centralized
10016 security network, a device accepts an initial active network key and updated network keys only from its
10017 Trust Center when secured with its Trust Center Link key. For purposes of configuration, a device accepts
10018 link keys intended for establishing end-to-end security between two devices only from its Trust Center or
10019 through application level negotiation using a higher level protocol between the two devices. Aside from the
10020 initial Trust Center link key or network key, additional link, and network keys are only accepted if they
10021 originate from a device’s Trust Center via secured key transport or negotiated using higher level application
10022 protocols.

10023 **4.3 NWK Layer Security**

10024 The NWK layer is responsible for the processing steps needed to securely transmit outgoing frames and
10025 securely receive incoming frames. Upper layers control the security processing operations by setting up the
10026 appropriate keys and frame counters and establishing which security level to use.

10027 **4.3.1 Frame Security**

10028 The detailed steps involved in security processing of outgoing and incoming NWK frames are described in
10029 sections 4.3.1.1 and 4.3.1.2, respectively.

4.3.1.1 Security Processing of Outgoing Frames

If the NWK layer has a frame, consisting of a header *NwkHeader* and payload *Payload*, which needs security protection and *nwkSecurityLevel* > 0, and in the case of a NWK data frame, the SecurityEnabled parameter in NLDEDEDATA.request had a value of TRUE, it shall apply security as follows:

1. Obtain the *nwkActiveKeySeqNumber* from the NIB and use it to retrieve the active network key *key*, outgoing frame counter *OutgoingFrameCounter*, and key sequence number *KeySeqNumber* from the *nwkSecurityMaterialSet* attribute in the NIB. Obtain the security level from the *nwkSecurityLevel* attribute from the NIB. If the outgoing frame counter is equal to $2^{32}-1$, or if the key cannot be obtained, security processing shall fail and no further security processing shall be done on this frame.
2. Construct the auxiliary header *AuxiliaryHeader* (see section 4.5.1):
 - a. Set the security control field as follows:
 - i. The security level sub-field shall be the security level obtained from step 1.
 - ii. The key identifier sub-field shall be set to '01' (that is, the active network key).
 - iii. The extended nonce sub-field shall be set to 1.
 - b. Set the source address field to the 64-bit extended address of the local device.
 - c. Set the frame counter field to the outgoing frame counter from step 1.
 - d. Set the key sequence number field to the sequence number from step 1.
3. Execute the CCM mode encryption and authentication operation, as specified in Annex A, with the following instantiations:
 - a. Obtain the parameter *M* from Table 4.40 corresponding to the security level from step 1;
 - b. The bit string *Key* shall be the key obtained from step 1;
 - c. The nonce *N* shall be the 13-octet string constructed using the security control field from step a, the frame counter field from step d, and the source address field from step c (see section 4.5.2.2);
 - d. If the security level requires encryption, the octet string *a* shall be the string *NwkHeader* || *AuxiliaryHeader* and the octet string *m* shall be the string *Payload*. Otherwise, the octet string *a* shall be the string *NwkHeader* || *AuxiliaryHeader* || *Payload* and the octet string *m* shall be a string of length zero.
4. If the CCM mode invoked in step 3 outputs 'invalid', security processing shall fail and no further security processing shall be done on this frame.
5. Let *c* be the output from step 3. If the security level requires encryption, the secured outgoing frame shall be *NwkHeader* || *AuxiliaryHeader* || *c*, otherwise the secured outgoing frame shall be *NwkHeader* || *AuxiliaryHeader* || *Payload* || *c*.
6. If the secured outgoing frame size is greater than *aMaxMacFrameSize* security processing shall fail and no further security processing shall be done on this frame.
7. The outgoing frame counter from step 1 shall be incremented by one and stored in the *OutgoingFrameCounter* element of the network security material descriptor referenced by the *nwkActiveKeySeqNumber* in the NIB; that is, the outgoing frame counter value associated with the key used to protect the frame is updated.
8. The security level sub-field of the security control field shall be over-written by the 3-bit all-zero string '000'.

4.3.1.2 Security Processing of Incoming Frames

If the NWK layer receives a secured frame (consisting of a header *NwkHeader*, auxiliary header *AuxiliaryHeader*, and payload *SecuredPayload*) as indicated by the security sub-field of the NWK header frame control field, it shall perform security processing as follows:

- 10074 1. Determine the security level from the *nwkSecurityLevel* attribute of the NIB. Over-write the 3-bit secu-
10075 rity level sub-field of the security control field of the *AuxiliaryHeader* with this value. Determine the
10076 sequence number *SequenceNumber*, sender address *SenderAddress*, and received frame count *Re-
10077 ceivedFrameCount* from the auxiliary header *AuxiliaryHeader* (see section 4.5.1). If *ReceivedFrame-
10078 Counter* is equal to $2^{32}-1$, security processing shall indicate a failure to the next higher layer with a
10079 status of 'bad frame counter' and no further security processing shall be done on this frame.
- 10080 2. Obtain the appropriate security material (consisting of the key and other attributes) by matching *Se-
10081 quenceNumber* to the sequence number of any key in the *nwkSecurityMaterialSet* attribute in the NIB.
10082 If the security material cannot be obtained, security processing shall indicate a failure to the next high-
10083 er layer with a status of 'frame security failed' and no further security processing shall be done on this
10084 frame.
- 10085 3. If there is an incoming frame count *FrameCount* corresponding to *SenderAddress* from the security
10086 material obtained in step 2 and if *ReceivedFrameCount* is less than *FrameCount*, security processing
10087 shall indicate a failure to the next higher layer with a status of 'bad frame counter' and no further secu-
10088 rity processing shall be done on this frame.
- 10089 4. Execute the CCM mode decryption and authentication checking operation, as specified in section A.2,
10090 with the following instantiations:
 - 10091 a. The parameter *M* shall be obtained from Table 4.40 corresponding to the security level from step
10092 1.
 - 10093 b. The bit string *Key* shall be the key obtained from step 2.
 - 10094 c. The nonce *N* shall be the 13-octet string constructed using the security control, the frame counter,
10095 and the source address fields from *AuxiliaryHeader* (see section 4.5.2.2). Note that the security
10096 level subfield of the security control field has been overwritten in step 1 and now contains the
10097 value determined from the *nwkSecurityLevel* attribute from the NIB.
 - 10098 d. The octet string *SecuredPayload* shall be parsed as *Payload1 || Payload2*, where the rightmost
10099 string *Payload2* is an *M*-octet string. If this operation fails, security processing shall indicate a
10100 failure to the next higher layer with a status of 'frame security failed' and no further security pro-
10101 cessing shall be done on this frame.
 - 10102 e. If the security level requires decryption, the octet string *a* shall be the string *NwkHeader || Auxil-*
10103 *iaryHeader* and the octet string *c* shall be the string *SecuredPayload*. Otherwise, the octet string *a*
10104 shall be the string *NwkHeader || AuxiliaryHeader || Payload1* and the octet string *c* shall be the
10105 string *Payload2*.
- 10106 5. Return the results of the CCM operation:
 - 10107 a. If the CCM mode invoked in step 4 outputs 'invalid', security processing shall indicate a failure to
10108 the next higher layer with a status of 'frame security failed' and no further security processing shall
10109 be done on this frame.
 - 10110 b. Let *m* be the output of step 4. If the security level requires encryption, set the octet string *Unse-
10111 curedNwkFrame* to the string *NwkHeader || m*. Otherwise, set the octet string *Unsecured-*
10112 *NwkFrame* to the string *NwkHeader || Payload1*.
- 10113 6. Set *FrameCount* to (*ReceivedFrameCount* + 1) and store both *FrameCount* and *SenderAddress* in the
10114 NIB. If storing this frame count and address information will cause the memory allocation for this
10115 type of information to be exceeded, and the *nwkAllFresh* attribute in the NIB is TRUE, then security
10116 processing shall fail and no further security processing shall be done on this frame. *Unsecured-*
10117 *NwkFrame* now represents the unsecured received network frame and security processing shall suc-
10118 ceed. So as to never cause the storage of the frame count and address information to exceed the availa-
10119 ble memory, the memory allocated for incoming frame counters needed for NWK layer security shall
10120 be bounded by *M*N*, where *M* and *N* represent the cardinality of *nwkSecurityMaterialSet* and
10121 *nwkNeighborTable* in the NIB, respectively.
- 10122 7. If the sequence number of the received frame belongs to a newer entry in the *nwkSecurityMaterialSet*,
10123 set the *nwkActiveKeySeqNumber* to the received sequence number.

- 10124 8. If there is an entry in nwkNeighborTable in the NIB whose extended address matches SenderAddress
 10125 and whose relationship field has value 0x05 (unauthenticated child), then set relationship field in that
 10126 entry to the value 0x01 (child).

10127 4.3.2 Secured NPDU Frame

10128 The NWK layer frame format (see section 3.3.1) consists of a NWK header and NWK payload field. The
 10129 NWK header consists of frame control and routing fields. When security is applied to an NPDU frame, the
 10130 security bit in the NWK frame control field shall be set to 1 to indicate the presence of the auxiliary frame
 10131 header. The format for the auxiliary frame header is given in section 4.5.1. The format of a secured NWK
 10132 layer frame is shown in Figure 4.3. The auxiliary frame header is situated between the NWK header and
 10133 payload fields.

10134 **Figure 4.3 Secured NWK Layer Frame Format**

Octets: Variable	14	Variable	
Original NWK header ([B3], Clause 7.1)	Auxiliary frame header	Encrypted payload	Encrypted message integrity code (MIC)
		Secure frame payload = output of CCM	
Full NWK header		Secured NWK payload	

10135 4.3.3 Security-Related NIB Attributes

10136 The NWK PIB contains attributes that are required to manage security for the NWK layer. Each of these
 10137 attributes can be read and written using the NLMEGET.request and NLME-SET.request primitives, respec-
 10138 tively. The security-related attributes contained in the NWK PIB are presented in Table 4.2, Table 4.3, and
 10139 Table 4.4.

10140 **Table 4.2 NIB Security Attributes**

Attribute	Identifier	Type	Range	Description	Default
<i>nwkSecurityLevel</i>	0xa0	Octet	0x00-07	The security level for out- going and incoming NWK frames; the allowable sec- urity level identifiers are pre- sented in Table 4.40.	0x05
<i>nwkSecurityMaterialSet</i>	0xa1	A set of 2 network security ma- terial de- scriptors (see Table 4.2)	Variable	Set of network security ma- terial descriptors capable of maintaining an active and alternate network key.	-

Attribute	Identifier	Type	Range	Description	Default
<i>nwkActiveKeySeqNumber</i>	0xa2	Octet	0x00-0xFF	The sequence number of the active network key in <i>nwkSecurityMaterialSet</i> .	0x00
<i>nwkAllFresh</i>	0xa3	Boolean	TRUE FALSE	Indicates whether incoming NWK frames must be all checked for freshness when the memory for incoming frame counts is exceeded. See section 4.3.1.2.	TRUE

10141

10142

Table 4.3 Elements of the Network Security Material Descriptor

Name	Type	Range	Description	Default
KeySeqNumber	Octet	0x00-0xFF	A sequence number assigned to a network key by the Trust Center and used to distinguish network keys for purposes of key updates, and incoming frame security operations. This is only used when operating in a centralized security network.	00
OutgoingFrameCounter	Ordered set of 4 octets.	0x00000000-0xFFFFFFFF	Outgoing frame counter used for outgoing frames.	0x00000000
IncomingFrameCounterSet	Set of incoming frame counter descriptor values. See Table 4.3.	Variable	Set of incoming frame counter values and corresponding device addresses.	Null set
Key	Ordered set of 16 octets.	-	The actual value of the key.	-

Name	Type	Range	Description	Default
NetworkKeyType	Octet	0x01 - 0x01	The type of the key. 0x01 = standard All other values are reserved.	0x01

10143

10144

Table 4.4 Elements of the Incoming Frame Counter Descriptor

Name	Type	Range	Description	Default
SenderAddress	Device address	Any valid 64-bit address	Extended device address.	Device specific
IncomingFrame Counter	Ordered set of 4 octets	0x00000000-0xFFFFFFFF	Incoming frame counter used for incoming frames.	0x00000000

10145

4.3.4 Network Frame Counter Requirements

10146
10147
10148
10149

Device shall maintain outgoing NWK frame counters across factory resets. The outgoing NWK frame counter must only be reset as detailed in this specification. A factory reset includes any over the air message, such as a NWK leave. It is permitted for manufacturers to provide a full factory reset that erases all persisted data as a separate user action.

10150
10151
10152
10153
10154
10155
10156
10157
10158
10159
10160
10161

A device can join a network, join other networks and then attempt to join the original network again. Neighbors on the original network will have a neighbor table entry for the device with the incoming frame counter set to the value that was heard when the device was previously on the network. If a fresh security material set with an outgoing NWK frame counter of zero is created when the original network is joined for a second time, devices in that network will reject frames sent with this frame counter. Devices must therefore have sufficient shadow copies of their security material set and associated EPID to store the outgoing frame counter and EPID for each network that they may join. As an implementation optimization, it is permissible to store a single instance of the outgoing NWK frame counter that is used across all security material sets. This outgoing NWK frame counter must be preserved across factory resets and when joining different networks. The only time the outgoing frame counter is reset to zero is when the device is already on a network, it receives an APSME-SWITCH-KEY and its outgoing frame counter is greater than 0x80000000.

10162

4.3.4.1 Network Frame Counter Usage Calculations

10163
10164
10165

One leap year is $366 * 24 * 60 * 60 = 31,622,400$ seconds. The frame counter will wrap every 4,294,967,295 counts. Therefore a device would need to continuously send at a rate greater than 135 packets per second to cause the frame counter to wrap in less than a year.

10166
10167
10168
10169

Often devices do not store the exact frame counter in flash memory but use a store ahead method to prevent wearing out flash memory. This will cause the device to jump its frame counter ahead on reboot to the next higher increment. If a device increments its frame counter by 1024 on a reboot, it would have to reboot at a rate greater than once every 7 seconds to cause a wrap in a year.

10170 A device must be able to store two network keys. If there are two network key updates whilst the device is
10171 asleep or turned off, it will no longer have a valid network key and will only be able to join the network via
10172 a Trust center rejoin. Limiting the network key updates to a maximum of once every 30 days mitigates this
10173 issue.

10174

10175 **4.4 APS Layer Security**

10176 The APS layer is responsible for the processing steps needed to securely transmit outgoing frames, securely
10177 receive incoming frames, and securely establish and manage cryptographic keys. Upper layers control the
10178 management of cryptographic keys by issuing primitives to the APS layer.

10179 Table 4.5 lists the primitives available for key management and maintenance. Upper layers also determine
10180 which security level to use when protecting outgoing frames.

10181 **Table 4.5 The APS Layer Security Primitives**

APSME Security Primitives	Request	Confirm	Indication	Response	Description
APSME-TRANSPORT-KEY	section 4.4.2.1	-	section 4.4.2.2	-	Transports security material from one device to another.
APSME- UPDATE-DEVICE	section 4.4.3.1	-	section 4.4.3.2	-	Notifies the Trust Center when a new device has joined, or an existing device has left the network.
APSME- REMOVE-DEVICE	section 4.4.4.1	-	section 4.4.4.2	-	Used by the Trust Center to notify a router that one of the router's child devices, or the router itself, should be removed from the network.
APSME-REQUEST-KEY	section 4.4.5.1	-	section 4.4.5.2	-	Used by a device to request that the Trust Center send an application link key or trust center link key.
APSME- SWITCH-KEY	section 4.4.6.1	-	section 4.4.6.2	-	Used by the Trust Center to tell a device to switch to a new network key.
APSME-VERIFY-KEY	section 4.4.7.1	-	section 4.4.7.2	-	Used by a device to verify the link key used by the trust center.

APSME Security Primitives	Request	Confirm	Indication	Response	Description
APSME-CONFIRM-KEY	section 4.4.8.1		section 4.4.8.2	-	Used by the trust center to confirm a previous request to verify a link key.

10182 **4.4.1 Frame Security**

10183 The detailed steps involved in security processing of outgoing and incoming APS frames are described in
10184 sections 4.4.1.1 and 4.4.1.2, respectively.

10185 **4.4.1.1 Security Processing of Outgoing Frames**

10186 If the APS layer has a frame, consisting of a header *ApsHeader* and payload *Payload*, that needs security
10187 protection and *nwkSecurityLevel* > 0, it shall apply security as follows:

- 10188 1. Obtain the security material and key identifier *KeyIdentifier* using the following procedure. If security
10189 material or key identifier cannot be determined, then security processing shall fail and no further secu-
10190 rity processing shall be done on this frame.
 - 10191 a. If the frame is a result of a APSDE-DATA.request primitive:
 - 10192 i. The security material associated with the destination address of the outgoing frame shall be
10193 obtained from the *apsDeviceKeyPairSet* attribute in the AIB. *KeyIdentifier* shall be set to '00'
10194 (that is, a data key).
 - 10195 ii. Only entries with a KeyAttribute of PROVISIONAL or VERIFIED shall be used. Keys
10196 with other attributes shall not be used for encryption.
 - 10197 b. If the frame is a result of an APS command that requires securing.
 - 10198 i. An attempt shall be made to retrieve the security material associated with the destination ad-
10199 dress of the outgoing frame from the *apsDeviceKeyPairSet* attribute in the AIB. Only en-
10200 tries with a KeyAttribute of PROVISIONAL or VERIFIED shall be used. Keys with other
10201 attributes shall not be used for encryption.
 - 10202 ii. For all cases except transport-key commands, *KeyIdentifier* shall be set to '00'(that is, a data
10203 key). For the case of transport-key commands, *KeyIdentifier* shall be set to '02' (that is, the
10204 key-transport key) when transporting a network key and shall be set to '03' (that is, the
10205 key-load key) when transporting an application link key or trust center link key. See section
10206 4.5.3 for a description of the key-transport and key-load keys.
- 10207 2. Extract the outgoing frame counter (and, if *KeyIdentifier* is 01, the key sequence number) from the se-
10208 curity material obtained from step 1. If the outgoing frame counter value is equal to integer $2^{32}-1$, or if
10209 the key cannot be obtained, security processing shall fail and no further security processing shall be
10210 done on this frame.
- 10211 3. Obtain the security level from the *nwkSecurityLevel* attribute from the NIB.
- 10212 4. Construct auxiliary header *AuxiliaryHeader* (see section 4.5.1). The security control field shall be set
10213 as follows:
 - 10214 a. The security level sub-field shall be the security level obtained from step 3.
 - 10215 i. The key identifier sub-field shall be set to *KeyIdentifier*.
 - 10216 ii. The extended nonce sub-field shall be set as follows: If the *ApsHeader* indicates the
10217 frame type is an APS Command, then the extended nonce sub-field shall be set to 1.
10218 Otherwise if the TxOptions bit for include extended nonce is set (0x10) then the extended
10219 nonce sub-field shall be set to 1. Otherwise it shall be set to 0.

- 10220 b. If the extended nonce sub-field is set to 1, then set the source address field to the 64-bit extended
10221 address of the local device.
10222 c. The frame counter field shall be set to the outgoing frame counter from step 2.
10223 d. If *KeyIdentifier* is ‘01’, the key sequence number field shall be present and set to the key sequence
10224 number from step 3. Otherwise, the key sequence number field shall not be present.
10225 5. Execute the CCM mode encryption and authentication operation, as specified in section A.2, with the
10226 following exceptions:
10227 a. The parameter *M* shall be obtained from Table 4.40 corresponding to the security level from step
10228 3.
10229 b. The bit string *Key* shall be the key obtained from step 1.
10230 c. The nonce *N* shall be the 13-octet string constructed using the security control and frame counter
10231 fields from step 5 and the 64-bit extended address of the local device (see section 4.5.2.2).
10232 d. If the security level requires encryption, the octet string *a* shall be the string *ApsHeader* || *AuxiliaryHeader*
10233 and the octet string *m* shall be the string *Payload*. Otherwise, the octet string *a* shall be
10234 the string *ApsHeader* || *AuxiliaryHeader* || *Payload* and the octet string *m* shall be a string of length
10235 zero.
10236 6. If the CCM mode invoked in step 6 outputs “invalid”, security processing shall fail and no further se-
10237 curity processing shall be done on this frame.
10238 7. Let *c* be the output from step 6. If the security level requires encryption, the secured outgoing frame
10239 shall be *ApsHeader* || *AuxiliaryHeader* || *c*, otherwise the secured outgoing frame shall be *ApsHeader* ||
10240 *AuxiliaryHeader* || *Payload* || *c*.
10241 8. If the secured outgoing frame size will result in the MSDU being greater than *aMaxMACFrameSize*
10242 octets (see IEEE Std. 802.15.4 802 [B1]), security processing shall fail and no further security pro-
10243 cessing shall be done on this frame.
10244 9. The outgoing frame counter from step 3 shall be incremented and stored in the appropriate location(s)
10245 of the NIB, AIB, and MAC PIB corresponding to the key that was used to protect the outgoing frame.
10246 10. Over-write the security level sub-field of the security control field with the 3- bit all-zero string ‘000’.

10247 **4.4.1.2 Security Processing of Incoming Frames**

10248 If the APS layer receives a secured frame (consisting of a header *ApsHeader*, auxiliary header *Auxiliary-
10249 Header*, and payload *SecuredPayload*) as indicated by the security sub-field of the APS header frame con-
10250 trol field it shall perform security processing as follows:

- 10251 1. Determine the sequence number *SequenceNumber*, key identifier *KeyIdentifier*, and received frame
10252 counter value *ReceivedFrameCounter* from the auxiliary header *AuxiliaryHeader*. If *ReceivedFrame-
10253 Counter* is the 4-octet representation of the integer $2^{32}-1$, security processing shall fail and no further
10254 security processing shall be done on this frame.
10255 2. Determine the source address *SourceAddress* from the address-map table in the NIB, using the source
10256 address in the APS frame as the index. If the source address is incomplete or unavailable, determine if
10257 the device is joined and unauthorized. If joined and unauthorized it shall use the *apsDeviceKeyPair-
10258 Set* that corresponds to its pre-installed link key. Otherwise, security processing shall fail and no fur-
10259 ther security processing shall be done on this frame.
10260 3. Obtain the appropriate security material in the following manner. If the security material cannot be ob-
10261 tained, security processing shall fail and no further security processing shall be done on this frame.
10262 a. If *KeyIdentifier* is ‘00’ (*i.e.*, a data key), the security material associated with the *SourceAddress* of
10263 the incoming frame shall be obtained from the *apsDeviceKeyPairSet* attribute in the AIB.
10264 b. If *KeyIdentifier* is ‘02’ (*i.e.*, a key-transport key), the security material associated with the *SourceAddress* of
10265 the incoming frame shall be obtained from the *apsDeviceKeyPairSet* attribute in the AIB; the key

- 10266 for this operation shall be derived from the security material as specified in section 4.5.3 for the
10267 key-transport key.
- 10268 c. If *KeyIdentifier* is ‘03’ (i.e., a key-load key), the security material associated with the *SourceAddress*
10269 of the incoming frame shall be obtained from the *apsDeviceKeyPairSet* attribute in the AIB
10270 and the key for this operation shall be derived from the security material as specified in section
10271 4.5.3 for the key-load key.
- 10272 4. If the *apsLinkKeyType* of the associated link key is 0x00 (unique) and there is an incoming frame count
10273 *FrameCount* corresponding to *SourceAddress* from the security material obtained in step 3 and if *ReceivedFrameCount*
10274 is less than *FrameCount*, security processing shall fail and no further security pro-
10275 cessing shall be done on this frame.
- 10276 5. Determine the security level *SecLevel* as follows. If the frame type sub-field of the frame control field
10277 of *ApsHeader* indicates an APS data frame, then *SecLevel* shall be set to the *nwkSecurityLevel* attribute
10278 in the NIB. Overwrite the security level sub-field of the security control field in the *AuxiliaryHeader*
10279 with the value of *SecLevel*.
- 10280 6. Execute the CCM mode decryption and authentication checking operation as specified in section A.3,
10281 with the following instantiations:
- 10282 a. The parameter *M* shall be obtained from Table 4.40 corresponding to the security level from step
10283 5.
- 10284 i. The bit string *Key* shall be the key obtained from step 3.
- 10285 ii. The nonce *N* shall be the 13-octet string constructed using the security control and frame
10286 counter fields from *AuxiliaryHeader*, and *SourceAddress* from step 2 (see section 4.5.2.2).
- 10287 iii. Parse the octet string *SecuredPayload* as *Payload1* || *Payload2*, where the rightmost
10288 string *Payload2* is an *M*-octet string. If this operation fails, security processing shall fail and no
10289 further security processing shall be done on this frame.
- 10290 iv. If the security level requires encryption, the octet string *a* shall be the string *ApsHeader* ||
10291 *AuxiliaryHeader* and the octet string *c* shall be the string *SecuredPayload*. Otherwise, the octet
10292 string *a* shall be the string *ApsHeader* || *AuxiliaryHeader* || *Payload1* and the octet string *c*
10293 shall be the string *Payload2*.
- 10294 7. Return the results of the CCM operation:
- 10295 a. If the CCM mode invoked in step 6 outputs “invalid”, security processing shall fail and no further
10296 security processing shall be done on this frame.
- 10297 b. Let *m* be the output of step 6. If the security level requires encryption, set the octet string *Unse-
10298 curedApsFrame* to the string *ApsHeader* || *m*. Otherwise, set the octet string *UnsecuredApsFrame*
10299 to the string *ApsHeader* || *Payload*.
- 10300 8. Set *FrameCount* to (*ReceivedFrameCount* + 1) and store both *FrameCount* and *SourceAddress* in the
10301 appropriate security material as obtained in step 3. If storing this frame count and address information
10302 will cause the memory allocation for this type of information to be exceeded, and the *nwkAllFresh* at-
10303 tribute in the NIB is TRUE, then security processing shall fail and no further security processing shall
10304 be done on this frame. Otherwise, security processing shall succeed.

10305

10306 4.4.1.3 Security Processing of APS Commands

10307 A device that is not the trust center that receives an APS command shall determine if the message was sent
10308 by the trust center or another device for which it has a link key. If operating in a centralized security
10309 network and the message was not sent by the trust center then it shall discard the message and no further
10310 processing shall be done.

10311 If operating in a centralized security network determining if the Trust Center sent the APS command shall
10312 be done as follows. If no APS encryption is present on the message then the device shall examine if there
10313 is an IEEE source address within the APS command frame. The IEEE source address shall be compared
10314 to the value of *apsTrustCenterAddress* in the AIB. If no IEEE source address is present in the APS com-
10315 mand frame then the device shall verify if the NWK source of the message is 0x0000. If there is APS en-
10316 cryption present on the APS command then the device shall verify that the key used to secure the message
10317 corresponds to the *apsDeviceKeyPairSet* that has a DeviceAddress equal to the value of the *apsTrustCen-*
10318 *terAddress* in the AIB.

10319 If the message was sent by the trust center the device shall then consult the AIB attribute *apsLinkKeyType*
10320 associated with the sending device to determine if the key is a unique link key or Global Link key. It shall
10321 then consult Table 4.6 to determine the policy that shall be used.

10322 **Table 4.6 Security Policy for Accepting APS Commands in a Centralized Security Network**

APS Command	Unique Trust Center Link Key (0x00)	Global Trust CenterLink Key (0x01)
Transport Key (0x05)	APS encryption is required as per device policy (see section 4.4.1.5).	APS encryption is required as per device policy (see section 4.4.1.5).
Update Device (0x06)	APS encryption required	APS encryption not required
Remove Device (0x07)	APS encryption required	APS encryption required
Request Key (0x08)	APS encryption required Trust Center Policy may further restrict, see section 4.4.1.5	APS encryption required Trust Center Policy may further restrict, see section 4.4.1.5
Switch Key (0x09)	APS encryption not required	APS encryption not required
Tunnel Data (0x0E)	APS encryption not required	APS encryption not required
Verify-Key (0x0F)	APS encryption not required.	APS encryption not required
Confirm-Key (0x10)	APS encryption required	APS encryption required.

10323
10324 Upon reception of an APS command that does not have APS encryption but APS encryption is required by
10325 Table 4.7, the device shall drop the message and no further processing shall take place. If APS encryption is
10326 not required for the command but the received message has APS encryption, the receiving device shall ac-
10327 cept and process the message. Accepting additional security on messages is required to support legacy de-
10328 vices in the field.

10329 In order to support backwards compatibility with devices in the field, provisions will also be added for new
10330 devices to ensure they can interoperate with the existing devices and their legacy requirements for APS en-
10331 cryption.

10332

Table 4.7 Security Policy for Sending APS Commands in a Centralized Security Network

APS Command	Unique Trust Center Link Key	Global Trust Center Link Key
Transport Key (0x05)	APS encryption may be optionally used. See section 4.4.1.4	APS encryption may be optionally used. See section 4.4.1.4
Update Device (0x06)	APS encryption shall be used.	APS encryption shall be conditionally used as per section 4.4.1.4.
Remove Device (0x07)	APS encryption shall be used	APS encryption shall be used
Request Key (0x08)	APS encryption shall be used	APS encryption shall be used
Switch Key (0x09)	APS encryption shall not be used	APS encryption shall not be used
Tunnel Data (0x0E)	APS encryption shall not be used	APS encryption shall not be used
Verify-Key (0x0F)	APS encryption shall not be used	APS encryption shall not be used
Confirm-Key (0x10)	APS encryption shall be used	APS encryption shall be used

10333

10334
10335
10336
10337
10338

When the local device will transmit an APS command, it shall consult Table 4.6 above to determine the appropriate behavior. If APS encryption is required to be used, then the device shall APS encrypt the command prior to sending the message. If APS encryption is not to be used, the device shall not APS encrypt the message prior to sending the message. Conditional encryption of APS commands shall follow the procedure as defined by section 4.4.1.4.

10339

4.4.1.4 Conditional Encryption of APS Commands

10340
10341

Devices may have requirements on when APS encryption must or must not be used. To ensure correct operation with those devices, the following procedure shall be undertaken as required by Table 4.6.

10342
10343
10344
10345
10346
10347

When sending an APS command that must be conditionally encrypted, the device shall send the APS command with APS encryption. If the receiving device is capable of accepting APS encrypted APS commands then the sending device may send APS encrypted APS commands. If the receiving device is not capable of receiving APS encrypted commands, then a response to the APS command will not be received. If the receiving device is not capable of receiving APS encrypted APS commands then the sending device can either not send the APS commands or send APS commands without APS encryption.

10348
10349
10350
10351

It is left up to the implementers to determine whether or not the receiving device is capable of receiving an APS command with APS encryption. A device may simply send two copies of the APS command, one with APS encryption and one without, in order to satisfy the requirements of interoperability with existing devices. Note this is not for APS datagrams this is for APS Command Frames.

10352
10353

Conditional encryption of APS commands shall only apply when the *apsLinkKeyType* with receiving device is set to Global Link key (0x01).

10354 **4.4.1.5 Acceptance of Commands Based on Security Policy**

10355 There are two commands that may be conditionally accepted based on the local security policies in place on
10356 the device.

10357 The APS transport key command may be sent with or without APS encryption. The decision to do so is
10358 based on the trust center's security policies. The trust center may deem it acceptable to send a key without
10359 APS encryption based on the method of transport.

10360 Conversely, a device receiving an APS transport key command may choose whether or not APS encryption
10361 is required. This is most often done during initial joining. For example, during joining a device that has no
10362 preconfigured link key would only accept unencrypted transport key messages, while a device with a pre-
10363 configured link key would only accept a transport key APS encrypted with its preconfigured key.

10364 The higher level specification implemented by the device may dictate the policies in place for these com-
10365 mands.

10366 A device that is in the joined and authorized state shall accept a broadcast NWK key update sent by the
10367 Trust Center using only NWK encryption. A device that is in state of joined and unauthorized shall re-
10368 quire an APS encrypted transport key if it has a preconfigured link key.

10369 **4.4.1.6 Conditional Encryption of APS Data**

10370 Devices and application profiles may have requirements on when APS encryption must or must not be used
10371 with normal APS Data. If the device has a set of application data encryption policies, then it shall encrypt
10372 any outgoing messages the policy indicates must be protected. It shall also reject any incoming messages
10373 that are not APS encrypted when the policy indicates encryption is required.

10374 If a device has requirements on encryption of APS data, it must establish application link keys with partner
10375 devices. In a centralized security network the trust center is used to broker this link key establishment.
10376 In a distributed security network the partner devices must establish a link key using an application defined
10377 method.

10378

10379 **4.4.2 Transport-Key Services**

10380 The APSME provides services that allow an initiator to transport keying material to a responder. The dif-
10381 ferent types of keying material that can be transported are shown in Table 4.14 to Table 4.17.

10382 **4.4.2.1 APSME-TRANSPORT-KEY.request**

10383 The APSME-TRANSPORT-KEY.request primitive is used for transporting a key to another device.

10384 **4.4.2.1.1 Semantics of the Service Primitive**

10385 This primitive shall provide the following interface:

10386 APSME-TRANSPORT-KEY.request	{
	DestAddress,
	StandardKeyType,
	TransportKeyData
	}

10391 Table 4.8 specifies the parameters for the APSME-TRANSPORT-KEY.request primitive.

10393

Table 4.8 APSME-TRANSPORT-KEY.request Parameters

Parameter Name	Type	Valid Range	Description
DestAddress	Device address	Any valid 64-bit address	The extended 64-bit address of the destination device.
StandardKeyType	Integer	0x00 – 0x06	Identifies the type of key material that should be transported; see Table 4.9.
TransportKeyData	Variable	Variable	The key being transported along with identification and usage parameters. The type of this parameter depends on the StandardKeyType parameter as follows: StandardKeyType = 0x01, Standard Network Key see Table 4.11 StandardKeyType = 0x03, Application Link Key see Table 4.12 StandardKeyType = 0x04, Trust Center Link Key, see Table 4.10

10394

10395

Table 4.9 StandardKeyType Parameter of the Transport-Key, Verify-Key, and Confirm-Key Primitives

Enumeration	Value	Description
Reserved	0x00	Reserved
Standard network key	0x01	Indicates that the key is a network key to be used in standard security mode
Reserved	0x02	Reserved
Application link key	0x03	Indicates the key is a link key used as a basis of security between two devices.
Trust-Center link key	0x04	Indicates that the key is a link key used as a basis for security between the Trust Center and another device.
Reserved	0x05 – 0xFF	Reserved

10396

10397

Table 4.10 TransportKeyData Parameter for a Trust Center Link Key

Parameter Name	Type	Valid Range	Description
Key	Set of 16 octets	Variable	The Trust Center link key.

10398

10399

Table 4.11 TransportKeyData Parameter for a Network Key

Parameter Name	Type	Valid Range	Description
KeySeqNumber	Octet	0x00-0xFF	A sequence number assigned to a network key by the Trust Center and used to distinguish network keys for purposes of key updates and incoming frame security operations.
NetworkKey	Set of 16 octets	Variable	The network key.
UseParent	Boolean	TRUE FALSE	This parameter indicates if the destination device's parent shall be used to forward the key to the destination device: TRUE = Use parent FALSE = Do not use parent
ParentAddress	Device address	Any valid 64-bit address	If the UseParent is TRUE, then ParentAddress parameter shall contain the extended 64-bit address of the destination device's parent device; otherwise, this parameter is not used and need not be set.

10400

10401

Table 4.12 TransportKeyData Parameter for an Application Link Key

Parameter Name	Type	Valid Range	Description
PartnerAddress	Device address	Any valid 64-bit address	The extended 64-bit address of the device that was also sent this link key.
Key	Set of 16 octets	Variable	The application link key

10402

4.4.2.1.2 When Generated

10403
10404

The ZDO on an initiator device shall generate this primitive when it requires a key to be transported to a responder device.

10405 **4.4.2.1.3 Effect on Receipt**

10406 The receipt of an APSME-TRANSPORT-KEY.request primitive shall cause the APSME to create a
10407 transport-key command packet (see section 4.4.9.2). If the StandardKeyType parameter is 0x04 (that is,
10408 Trust Center link key), the key descriptor field of the transport-key command shall be set as follows:

- 10409 • The key sub-field shall be set to the Key sub-parameter of the TransportKeyData parameter.
- 10410 • The destination address sub-field shall be set to the DestinationAddress parameter.
- 10411 • The source address sub-field shall be set to the local device address.

10412 This command frame shall be security-protected as specified in section 4.4.1. Then, if security processing
10413 succeeds, it is sent to the device specified by the ParentAddress sub-parameter of the TransportKeyData
10414 parameter by issuing a NLDE-DATA.request primitive.

10415 If the DestinationAddress parameter is all zeros, the secured command frame shall be unicast to any and all
10416 rx-off-when-idle children of the device. These unicasts shall be repeated until successful, or a subsequent
10417 APSME-TRANSPORT-KEY.request primitive with the StandardKeyType parameter equal to 0x01 has
10418 been received, or a period of twice the recommended maximum polling interval has passed.

10419 If the StandardKeyType parameter is 0x01 (that is, a network key), the key descriptor field of the
10420 transport-key command shall be set as follows:

- 10421 • The key sub-field shall be set to the Key sub-parameter of the TransportKeyData parameter.
- 10422 • The sequence number sub-field shall be set to the KeySeqNumber sub-parameter of the
10423 TransportKeyData parameter.
- 10424 • The destination address sub-field shall be set to the DestinationAddress parameter.
- 10425 • The source address sub-field shall be set to the local device address.

10426 This command frame shall be security-protected as specified in section 4.4.1.1 and then, if security pro-
10427 cessing succeeds, sent to the device specified by the ParentAddress sub-parameter of the TransportKeyData
10428 parameter (if the UseParent sub-parameter of the TransportKeyData parameter is TRUE) or the Destina-
10429 tionAddress parameter (if the UseParent sub-parameter of the TransportKeyData parameter is FALSE) by
10430 issuing a NLDE-DATA.request primitive.

10431 If the StandardKeyType parameter is 0x03 (that is, an application link key), the key descriptor field of the
10432 transport-key command shall be set as follows:

- 10433 • The key sub-field shall be set to the Key sub-parameter of the TransportKeyData parameter.
- 10434 • The partner address sub-field shall be set to the PartnerAddress sub-parameter of the Transport-
10435 KeyData parameter.
- 10436 • The initiator sub-field shall be set 1 (if the Initiator sub-parameter of the TransportKeyData pa-
10437 rameter is TRUE) or 0 (if the Initiator sub-parameter of the TransportKeyData parameter is
10438 FALSE).

10439 This command frame shall be security-protected as specified in sub-clause 4.4.1.1 and then, if security
10440 processing succeeds, sent to the device specified by the DestinationAddress parameter by issuing a
10441 NLDE-DATA.request primitive.

10442 **4.4.2.2 APSME-TRANSPORT-KEY.indication**

10443 The APSME-TRANSPORT-KEY.indication primitive is used to inform the ZDO of the receipt of keying
10444 material.

10445 **4.4.2.2.1 Semantics of the Service Primitive**

10446 This primitive shall provide the following interface:

10447 APSME-TRANSPORT-KEY.indication {
10448 SrcAddress,

10449 StandardKeyType,
10450 TransportKeyData
10451 }
10452
10453

Table 4.13 specifies the parameters of the APSME-TRANSPORT-KEY.indication primitive.

10454

Table 4.13 APSME-TRANSPORT-KEY.indication Parameters

Parameter Name	Type	Valid Range	Description
SrcAddress	Device Address	Any valid 64-bit address	The extended 64-bit address of the device that is the original source of the transported key.
StandardKeyType	Octet	0x00 – 0x06	Identifies the type of key material that was be transported; see Table 4.9.
TransportKeyData	Variable	Variable	<p>The key that was transported along with identification and usage parameters. The type of this parameter depends on the StandardKeyType parameter as follows:</p> <p>StandardKeyType = 0x01 see Table 4.11 StandardKeyType = 0x03 see Table 4.12 StandardKeyType = 0x04 see Table 4.10</p>

10455

4.4.2.2.2 When Generated

10456 The APSME shall generate this primitive when it receives a transport-key command as specified in section 4.4.3.3.

4.4.2.2.3 Effect on Receipt

10457 Upon receipt of this primitive, the ZDO is informed of the receipt of the keying material.

4.4.2.3 Upon Receipt of a Transport-Key Command

10458 Upon receipt of a transport-key command, the APSME shall execute security processing as specified in, then check the key type sub-field.

10459 Upon receipt of a secured transport-key command, the APSME shall check the key type sub-field. If the key type field is set to 0x03 or 0x04 (that is, application link or Trust Center link key) and the receiving device is operating in the joined and authorized state and the command was not secured using a distributed security link key or a Trust Center link key, the command shall be discarded. If the device is operating in the joined and authorized state it may accept a NWK broadcast transport key command with Key type field set to 0x01 (that is, network key) where the message has no APS encryption. If the key type field is set to 0x01 (that is, network key) and the command was not secured using a distributed security link key, Trust Center link key, the command shall be discarded.

10460 If the key type field is set to 0x03 (that is, application link key), the APSME shall issue the APSME-TRANSPORT-KEY.indication primitive with: the SrcAddress parameter set to the source of the key-transport command (as indicated by the NLDE-DATA.indication SrcAddress parameter), and the StandardKeyType parameter set to the key type field. The TransportKeyData parameter shall be set as follows:

- The Key sub-parameter shall be set to the key field.
- The PartnerAddress sub-parameter shall be set to the partner address field.

- 10479 • The Initiator parameter shall be set to TRUE, if the initiator field is 1. Otherwise it shall be set to
10480 0.

10481 If the key type field is set to 0x01 or 0x04, (that is, Trust Center link key, or a network key) and the destination
10482 field is equal to the local address, or if the key type field is set to 0x01 (that is, a network key), the destination
10483 field is the all-zero string, and the current network key type is equal to the value of the key type
10484 field, the APSME shall issue the APSME-TRANSPORT-KEY.indication primitive with the SrcAddress
10485 parameter set to the source address field of the key-transport command and the StandardKeyType parameter
10486 set to the key type field. The TransportKeyData parameter shall be set as follows: the Key
10487 sub-parameter shall be set to the key field and, in the case of a network key (that is, the key type field is set
10488 to 0x01), the KeySeqNumber sub-parameter shall be set to the sequence number field.

10489 If the key type field is set to 0x01 (network key) and source address field is set to 0xFFFFFFFFFFFFFF
10490 this indicates a distributed security network, the APSME shall issue the APSME-TRANSPORT-KEY.indication primitive with the SrcAddress parameter set to the source address
10491 field of the key-transport command and the StandardKeyType parameter set to the key type field. The TransportKeyData parameter shall be set as follows: the Key subparameter shall be set to the key field and,
10492 in the case of a network key (that is, the key type field is set to 0x01), the KeySeqNumber sub-parameter
10493 shall be set to the sequence number field. The *apsTrustCenterAddress* should be set to
10494 0xFFFFFFFFFFFFFF indicating a distributed security network.

10495 If the key type field is set to 0x04 or 0x01 (that is, Trust Center link key or network key) and the destination
10496 address field is not equal to the local address, the APSME shall send the command to the address indicated
10497 by the destination address field by issuing the NLDE-DATA.request primitive with security disabled.

10498 Upon receipt of a secured transport-key command with the key type field set to 0x01, if the destination
10499 field is all zeros and the source address field is set to the value of *apsTrustCenterAddress*, the router shall
10500 attempt to unicast this transport-key command to any and all rx-off-when-idle children. The router shall
10501 continue to do so until successful, or until a subsequent transport-key command with the key type field set
10502 to 0x01 or 0x05 has been received, or until a period of twice the recommended maximum polling interval
10503 has passed.

10504

4.4.3 Update Device Services

10505 The APSME provides services that allow a device (for example, a router) to inform another device (for
10506 example, a Trust Center) that a third device has changed its status (for example, joined or left the network).

4.4.3.1 APSME-UPDATE-DEVICE.request

10507 The APSME shall issue this primitive when it wants to inform a device (for example, a Trust Center) that
10508 another device has a status that needs to be updated (for example, the device joined or left the network).

4.4.3.1.1 Semantics of the Service Primitive

10509 This primitive shall provide the following interface:

10510 APSME-UPDATE-DEVICE.request {
10511 DestAddress,
10512 DeviceAddress,
10513 Status,
10514 DeviceShortAddress
10515 }

10516

10522 Table 4.13 specifies the parameters for the APSME-UPDATE-DEVICE.request primitive.

10523

10524 **Table 4.14 APSME-UPDATE-DEVICE.request Parameters**

Parameter Name	Type	Valid Range	Description
DestAddress	Device Address	Any valid 64-bit address	The extended 64-bit address of the device that shall be sent the update information.
DeviceAddress	Device Address	Any valid 64-bit address	The extended 64-bit address of the device whose status is being updated.
Status	Integer	0x00 – 0x07	Indicates the updated status of the device given by the DeviceAddress parameter: 0x00 = Standard device secured rejoin 0x01 = Standard device unsecured join 0x02 = Device left 0x03 = Standard device trust center rejoin 0x04 – 0x07 = Reserved
DeviceShortAddress	Network address	0x0000 - 0xffff	The 16-bit network address of the device whose status is being updated.

10525 **4.4.3.1.2 When Generated**

10526 The APSME (for example, on a router or ZigBee coordinator) shall initiate the
10527 APSME-UPDATE-DEVICE.request primitive when it wants to send updated device information to another
10528 device (for example, the Trust Center).

10529 **4.4.3.1.3 Effect on Receipt**

10530 Upon receipt of the APSME-UPDATE-DEVICE.request primitive, the device shall first create an update-device command frame (see section 4.4.9.3). The device address field of this command frame shall be
10531 set to the DeviceAddress parameter, the status field shall be set according to the Status parameter, and the
10532 device short address field shall be set to the DeviceShortAddress parameter. This command frame shall be
10533 security-protected as specified in section 4.4.1.1 and then, if security processing succeeds, sent to the de-
10534 vice specified in the DestAddress parameter by issuing a NLDE-DATA.request primitive.
10535

10536 **4.4.3.2 APSME-UPDATE-DEVICE.indication**

10537 This primitive is issued to inform the APSME that it received an update-device command frame.

10538 **4.4.3.2.1 Semantics of the Service Primitive**

10539 This primitive shall provide the following interface:

10540 APSME-UPDATE-DEVICE.indication {
10541 SrcAddress,
10542 DeviceAddress,

10543 Status,
10544 DeviceShortAddress
10545 }
10546
10547 Table 4.15 specifies the parameters for the APSME-UPDATE-DEVICE.indication primitive.
10548 **Table 4.15 APSME-UPDATE-DEVICE.indication Parameters**

Parameter Name	Type	Valid Range	Description
SrcAddress	Device Address	Any valid 64-bit address	The extended 64-bit address of the device originating the update-device command.
DeviceAddress	Device Address	Any valid 64-bit address	The extended 64-bit address of the device whose status is being updated.
Status	Integer	0x00 – 0x07	Indicates the updated status of the device given by the DeviceAddress parameter: 0x00 = Standard device secured rejoin 0x01 = Standard device unsecured join 0x02 = Device left 0x03 = Standard device trust center rejoin 0x04 – 0x07 = Reserved
DeviceShortAddress	Network Address	0x0000-0xffff	The 16-bit network address of the device whose status is being updated.

10549 **4.4.3.2.2 When Generated**

10550 The APSME shall generate this primitive when it receives an update-device command frame that is successfully decrypted and authenticated, as specified in section 4.4.1.2.
10551

10552 **4.4.3.2.3 Effect on Receipt**

10553 Upon receipt of the APSME-UPDATE-DEVICE.indication primitive, the APSME will be informed that
10554 the device referenced by the DeviceAddress parameter has undergone a status update according to the Status
10555 parameter.

10556 **4.4.4 Remove Device Services**

10557 The APSME provides services that allow a device (for example, a Trust Center) to inform another device
10558 (for example, a router) that one of its children should be removed from the network.

10559 These services may be used in distributed network security.

4.4.4.1 APSME-REMOVE-DEVICE.request

10561 The APSME of a device (for example, a Trust Center) shall issue this primitive when it wants to request
10562 that a parent device (for example, a router) remove one of its children from the network. For example, a
10563 Trust Center can use this primitive to remove a child device that is not authorized to be on the network.

4.4.4.1.1 Semantics of the Service Primitive

10565 This primitive shall provide the following interface:

10566 APSME-REMOVE-DEVICE.request {
10567 ParentAddress,
10568 ChildAddress
10569 }

10570
10571 Table 4.16 specifies the parameters for the APSME-REMOVE-DEVICE.request primitive.
10572

Table 4.16 APSME-REMOVE-DEVICE.request Parameters

Parameter Name	Type	Valid Range	Description
ParentAddress	Device Address	Any valid 64-bit address	The extended 64-bit address of the device that is the parent of the child device that is requested to be removed, or the router device that is requested to be removed.
TargetAddress	Device Address	Any valid 64-bit address	The extended 64-bit address of the target device that is requested to be removed. If a router device is requested to be removed, then the <i>ParentAddress</i> shall be the same as the <i>TargetAddress</i> .

4.4.4.1.2 When Generated

10573
10574 The APSME (for example, on a Trust Center) shall initiate the APSME-REMOVE-DEVICE.request primitive
10575 when it wants to request that a parent device (specified by the ParentAddress parameter) remove one of
10576 its child devices (as specified by the TargetAddress parameter), or if it wants to remove a router from the
10577 network.

10578 If the device being removed is a router then the ParentAddress field shall be set to the EUI64 of that router
10579 and the TargetAddress shall be set to the same value.

4.4.4.1.3 Effect on Receipt

10581 Upon receipt of the APSME-REMOVE-DEVICE.request primitive the device shall first create a re-
10582 move-device command frame (see section 4.4.9.3). The address field of this command frame shall be set to
10583 the TargetAddress parameter. If the device to be removed is a router the ParentAddress and TargetAddress
10584 shall be the same. This command frame shall be security-protected as specified in section 4.4.1.1 and
10585 then, if security processing succeeds, sent to the device specified by the ParentAddress parameter by issuing
10586 a NLDE-DATA.request primitive.

4.4.4.2 APSME-REMOVE-DEVICE.indication

10587 The APSME shall issue this primitive to inform the ZDO that it received a remove-device command frame.
10588

10589 **4.4.4.2.1 Semantics of the Service Primitive**

10590 This primitive shall provide the following interface:

10591 APSME-REMOVE-DEVICE.indication {
10592 SrcAddress,
10593 ChildAddress
10594 }

10595
10596 Table 4.17 specifies the parameters for the APSME-REMOVEDEVICE.indication primitive.

10597 **Table 4.17 APSME-REMOVE-DEVICE.indication Parameters**

Parameter Name	Type	Valid Range	Description
SrcAddress	Device Address	Any valid 64-bit address	The extended 64-bit address of the device requesting that a child device be removed.
TargetAddress	Device Address	Any valid 64-bit address	The extended 64-bit address of the target device that is requested to be removed.

10598
10599 **4.4.4.2.2 When Generated**

10600 The APSME shall generate this primitive when it receives a remove-device command frame that is successfully decrypted and authenticated, as specified in section 4.4.1.2.

10602 **4.4.4.2.3 Effect on Receipt**

10603 Upon receipt of the APSME-REMOVE-DEVICE.indication primitive the ZDO shall be informed that the device referenced by the TargetAddress parameter shall be removed from the network.

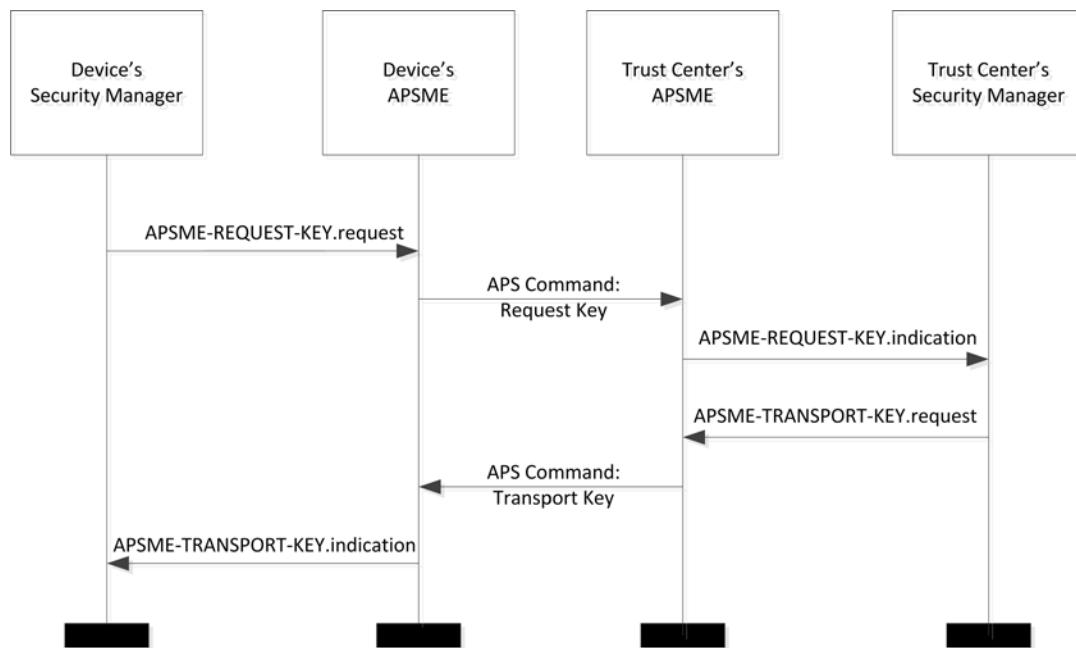
10605 It shall generate an NLME-LEAVE.request and process it as described in 3.2.2.16.

10606 **4.4.5 Request Key Services**

10607 The APSME provides services that allow a non-trust center device to request an application or trust center link key from the Trust Center. Figure 4.4 shows the processing for the request key services.

10610

Figure 4.4 Request Key Service Processing for Trust Center Link Key



10611

4.4.5.1 APSME-REQUEST-KEY.request

10612 This primitive allows the Security Manager to request a new trust center link key or a new end-to-end application link key.

4.4.5.1.1 Semantics of the Service Primitive

10613 This primitive shall provide the following interface:

10617 APSME-REQUEST-KEY.request	{
10618 DestAddress,	
10619 RequestKeyType,	
10620 PartnerAddress	
10621 }	

10622

10623 Table 4.18 specifies the parameters for the APSME-REQUEST-KEY.request primitive.

10624 **Table 4.18 APSME-REQUEST-KEY.request Parameters**

Parameter Name	Type	Valid Range	Description
DestAddress	Device Address	Any valid 64-bit address	The extended 64-bit address of the device to which the request-key command should be sent.

RequestKeyType	Octet	0x02 and 0x04	The type of key being requested. See Table 4.19.
PartnerAddress	Device Address	Any valid 64-bit address	If the RequestKeyType parameter indicates an application key, this parameter shall indicate an extended 64-bit address of a device that shall receive the same key as the device requesting the key.

10625 Table 4.19 describes the values of the RequestKeyType enumeration. Please note that this enumeration is
10626 different than the one for the StandardKeyType in Table 4.9.

10627

10628

Table 4.19 RequestKeyType Values

Value	Enumeration
0x00	Reserved
0x01	Reserved
0x02	Application Link Key
0x03	Reserved
0x04	Trust Center Link Key
0x05 – 0xFF	Reserved

10629

4.4.5.1.2 When Generated

10630 The Security Manager of a device shall generate the APSME-REQUEST-KEY.request primitive when it
10631 requires either a new end-to-end application link key or trust center link key. An application link key with
10632 the Trust Center is also known as a Trust Center Link Key.

10633

4.4.5.1.3 Effect on Receipt

10634 Upon receipt of the APSME-REQUEST-KEY.request primitive, the device shall first create an APS request-key command frame (see section 4.4.9.5). The RequestKeyType field of this command frame shall be set to the same value as the RequestKeyType parameter. If the RequestKeyType parameter is 0x02 (that is, an application link key), then the partner address field of this command frame shall be the PartnerAddress parameter. Otherwise, the partner address field of this command frame shall not be present.

10640 This command frame shall be security-protected as specified in section 4.4.1.1 and then, if security pro-
10641 cessing succeeds, sent to the device specified by the DestAddress parameter by issuing a
10642 NLDE-DATA.request primitive.

4.4.5.2 APSME-REQUEST-KEY.indication

10644 The APSME shall issue this primitive to inform the Security Manager that it received a request-key com-
10645 mand frame.

4.4.5.2.1 Semantics of the Service Primitive

10647 This primitive shall provide the following interface:

10648 APSME-REQUEST-KEY.indication {
10649 SrcAddress,
10650 RequestKeyType,
10651 PartnerAddress
10652 }

10653 Table 4.20 specifies the parameters for the APSME-REQUEST-KEY.indication primitive.

10654 **Table 4.20 APSME-REQUEST-KEY.indication Parameters**

Parameter Name	Type	Valid Range	Description
SrcAddress	Device Address	Any valid 64-bit address	The extended 64-bit address of the device that sent the request-key command.
RequestKeyType	Octet	See Description.	The type of key being requested. See Table 4.19 for a list of types and valid values.
PartnerAddress	Device Address	Any valid 64-bit address	If the RequestKeyType parameter indicates an application key, this parameter shall indicate an extended 64-bit address of a device that shall receive the same key as the device requesting the key.

4.4.5.2.2 When Generated

10657 The APSME shall generate this primitive when it receives a request-key command frame that is success-
10658 fully decrypted and authenticated, as specified in section 4.4.1.2.

4.4.5.2.3 Effect on Receipt

10660 Upon receipt of the APSME-REQUEST-KEY.indication primitive, the following shall be done:

1. If the device is not the Trust Center, the request shall be silently dropped and no further processing shall take place.
2. If the apsTrustCenterAddress of the AIB is 0xFFFFFFFFFFFFFF (indicating a distributed security network), the request shall be silently dropped and no further processing shall take place.

- 10665 3. If the RequestKeyType is 0x04, Trust Center Link Key, then follow the procedure in section
10666 4.7.3.6.
10667 4. If the RequestKeyType is 0x02, Application Link Key, then follow the procedure in section
10668 4.7.3.8.
10669 5. If the RequestKeyType is any other value, the request shall be silently dropped and no further
10670 processing shall take place.

4.4.6 Switch Key Services

10672 The APSME provides services that allow the Trust Center to inform another device that it should switch to
10673 a new active network key.

4.4.6.1 APSME-SWITCH-KEY.request

10675 This primitive allows a device (for example, the Trust Center) to request that another device or all devices
10676 switch to a new active network key.

4.4.6.1.1 Semantics of the Service Primitive

10678 This primitive shall provide the following interface:

```
APSME-SWITCH-KEY.request {  
    DestAddress,  
    KeySeqNumber  
}
```

10683 Table 4.21 specifies the parameters for the APSME-SWITCH-KEY.request primitive.

10684 **Table 4.21 APSME-SWITCH-KEY.request Parameters**

Parameter Name	Type	Valid Range	Description
DestAddress	Device Address	Any valid 64-bit address	The extended 64-bit address of the device to which the switch-key command is sent. This may be the broadcast address 0xFFFFFFFFFFFFFF.
KeySeqNumber	Octet	0x00-0xFF	A sequence number assigned to a network key by the Trust Center and used to distinguish network keys.

4.4.6.1.2 When Generated

10686 The ZDO of a device (for example, the Trust Center) shall generate the APSME-SWITCH-KEY.request primitive when it wants to inform a device or all devices to switch to a new active network key.

4.4.6.1.3 Effect on Receipt

10687 Upon receipt of the APSME-SWITCH-KEY.request primitive, the device shall first create a switch-key command frame (see section 4.4.9.6). The sequence number field of this command frame shall be set to the same value as the KeySeqNumber parameter.

10694 If the DestAddress is not the broadcast address 0xFFFFFFFFFFFFFF, this command frame shall be security-protected as specified in section 4.4.1.1 and then, if security processing succeeds, sent to the device specified by the DestAddress parameter by issuing a NLDE-DATA.request primitive.

10697 If the DestAddress is the broadcast address 0xFFFFFFFFFFFFFF then the command shall not be security protected at the APS layer. It shall be sent to the NWK broadcast address 0xFFFF by issuing a NLDE-DATA.request primitive.

4.4.6.2 APSME-SWITCH-KEY.indication

The APSME shall issue this primitive to inform the ZDO that it received a switch-key command frame.

4.4.6.2.1 Semantics of the Service Primitive

This primitive shall provide the following interface:

10704

```
10705     APSME-SWITCH-KEY.indication {  
10706         SrcAddress,  
10707         KeySeqNumber  
10708     }
```

10709 Table 4.22 specifies the parameters for the APSME-SWITCH-KEY.indication primitive.

10710 **Table 4.22 APSME-SWITCH-KEY.indication Parameters**

Parameter Name	Type	Valid Range	Description
SrcAddress	Device Address	Any valid 64-bit address	The extended 64-bit address of the device that sent the switch-key command.
KeySeqNumber	Octet	0x00-0xFF	A sequence number assigned to a network key by the Trust Center and used to distinguish network keys.

10711 **4.4.6.2.2 When Generated**

10712 The APSME shall generate this primitive when it receives a switch-key command frame that is successfully
10713 decrypted and authenticated, as specified in section 4.4.1.2.

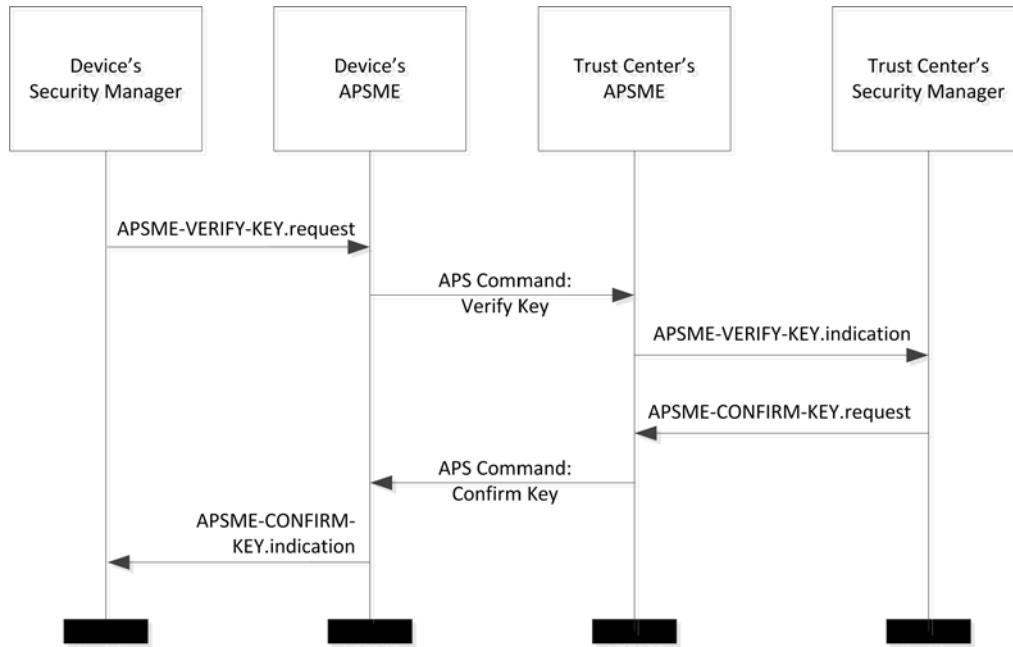
10714 **4.4.6.2.3 Effect on Receipt**

10715 Upon receipt of the APSME-SWITCH-KEY.indication primitive the ZDO shall be informed that the device
10716 referenced by the SrcAddress parameter is requesting that the network key referenced by the Key-
10717 SeqNumber parameter become the new active network key.

4.4.7 ¹Verify-Key Services

Figure 4.5 illustrates the flow of service requests and the over-the-air messages for the verify key.

Figure 4.5 Verify-Key Processing



4.4.7.1 APSME-VERIFY-KEY.request

This primitive allows a device to request that the partner device verify the Link Key between the two devices.

4.4.7.1.1 Semantics of the Service Primitive

The primitive shall provide the following interface:

```
APSME-VERIFY-KEY.request {  
    DestAddress,  
    StandardKeyType  
}
```

Table 4.23 specifies the parameters of the APSME-VERIFY-KEY.request primitive.

¹ Note: This is moved text. Moved to section 4.4.9.

10734

10735

Table 4.23 APSME-VERIFY-KEY.request Parameters

Parameter Name	Type	Valid Range	Description
DestAddress	Device Address	Any valid 64-bit address	The extended 64-bit address of the device to which the verify-key command be sent.
StandardKeyType	Octet	0x00-0xFF	Type of key being verified. See Table 4.9.

4.4.7.1.2 When Generated

The Security Manager on an initiator device shall generate this primitive when it wants to verify its Trust Center link key with the Trust Center.

4.4.7.1.3 Effect on Receipt

On receipt of the APSME-VERIFY-KEY.request primitive the following shall be performed:

1. If the local device is the Trust Center, the request is invalid and no further processing shall be done.
2. If the StandardKeyType parameter is not equal to 0x04 (Trust Center Link Key), the request is invalid. No further processing shall be done.
3. If the apsTrustCenterAddress of the AIB is 0xFFFFFFFFFFFFFF (indicating a distributed security network), then the request is invalid. No further processing shall be done.
4. If the DestAddress parameter is not equal to the apsTrustCenterAddress of the AIB, then the request is invalid. No further processing shall be done.
5. The device shall find the corresponding entry in the apsDeviceKeyPairSet that has a DeviceAddress equal to the apsTrustCenterAddress of AIB. If no entry can be found, the operation has failed and no further processing shall be done.
6. The *Initiator Verify-Key Hash Value* shall be calculated according to section 4.5.3 using the LinkKey value of the corresponding apsDeviceKeyPairSet entry found in step 5.
7. The APSME shall generate an APS Command Verify-Key setting the StandardKeyType in the command to the StandardKeyType of this primitive, and setting the Hash value to the calculated Initiator Verify-Key Hash Value. The APS command shall not be APS encrypted.

4.4.7.2 APSME-VERIFY-KEY.indication

This primitive allows a Trust Center to be notified when a device is requesting to verify its Trust Center Link Key. It allows the Trust Center to know when a provisional link key has been replaced by a verified link key.

4.4.7.2.1 Semantics of the Service Primitive

The primitive shall provide the following interface:

APSME-VERIFY-KEY.indication {

```

10766           SrcAddress,
10767           StandardKeyType,
10768           ReceivedInitiatorHashValue
10769       }

```

10770 Table 4.24 specifies the parameters of the APSME-VERIFY-KEY.indication primitive.

10771
10772 **Table 4.24 APSME-VERIFY-KEY.indication Parameters**

Parameter Name	Type	Valid Range	Description
SrcAddress	Device Address	Any valid 64-bit address	The extended 64-bit address of the device that sent the verify-key command.
StandardKeyType	Octet	0x00-0xFF	Type of key being verified. See Table 4.9.
ReceivedInitiatorHashValue	Set of 16 octets	Variable	The initiator hash of the key being verified.

10773 **4.4.7.2.2 When Generated**

10774 The APSME shall generate this primitive when it receives an APS Command Verify Key.

10775 **4.4.7.2.3 Effect on Receipt**

10776 On receipt of the APSME-VERIFY-KEY.indication primitive the following shall be performed:

1. If the message is a NWK broadcast, the request shall be dropped and no further processing shall be done.
2. If the device is not the Trust Center, this is not a valid request. The device shall follow the procedure in section 4.4.7.2.3.1 setting the Status value to 0xa3 (ILLEGAL_REQUEST). No further processing shall be done.
3. If the StandardKeyType parameter is not equal to 0x04 (Trust Center Link Key), the request is invalid. The device shall follow the procedure in section 4.4.7.2.3.1 setting the Status value to 0xaa (NOT_SUPPORTED). No further processing shall be done.
4. If the apsTrustCenterAddress of the AIB is set to 0xFFFFFFFFFFFFFF, the device is operating in distributed Trust Center mode and this is not a valid request. The device shall follow the procedure in section 4.4.7.2.3.1 setting the Status value to 0xaa (NOT_SUPPORTED). No further processing shall be done.
5. The device shall find the corresponding entry in the *apsDeviceKeyPairSet* attribute of the AIB where the DeviceAddress matches the SrcAddress of this primitive and the KeyAttributes is UN-VERIFIED_KEY (0x01) or VERIFIED_KEY (0x02). If no entry matching those criteria is found, the following shall be performed.
 - a. The Security Manager shall follow the procedure in section 4.4.7.2.3.1 setting the Status value to 0xad (SECURITY_FAILURE).
 - b. No further processing shall be done.

- 10796 6. The device shall calculate the CalculatedInitiatorHashValue by using the LinkKey value in the
10797 corresponding *apsDeviceKeyPairSet* entry and the *Initiator Verify-Key Hash Value* cryptographic
10798 operation described in section 4.5.3.
- 10799 7. The device shall compare the ReceivedInitiatorHashValue of the primitive with the CalculatedInitiatorHashValue. If the values do not match the operation has failed, the following shall be performed.
- 10802 a. The Security Manager shall follow the procedure in section 4.4.7.2.3.1 setting the Status
10803 value to 0xad (SECURITY_FAILURE).
- 10804 b. No further processing shall be done.
- 10805 8. The device shall set the KeyAttributes of the corresponding *apsDeviceKeyPairSet* entry to VERI-
10806 FIED_KEY (0x02).
- 10807 9. The device shall attempt to find the entry in the *apsDeviceKeyPairSet* where the DeviceAddress of
10808 the entry matches the SrcAddress of this primitive and the KeyAttributes is set to PROVISION-
10809 AL_KEY (0x00).
- 10810 a. If an entry is found, that entry shall be deleted from the *apsDeviceKeyPairSet*. Processing
10811 shall continue.
- 10812 b. If no entry is found, then processing shall continue.
- 10813 10. The device shall follow the procedure in section 4.4.7.2.3.1 setting the Status value to 0x00
10814 (SUCCESS).

10816 **4.4.7.2.3.1 APSME-VERIFY-KEY.indication Response**

10817 The following shall be done when an APSME-VERIFY-KEY.indication indicates a response must be generated. This procedure takes a Status code as a parameter.

10819 An APSME-CONFIRM-KEY.request shall be generated with the following values:

- 10820 1. The Status code shall be set to the Status code passed to this procedure.
- 10821 2. The DestAddress shall be set to the SrcAddress of the APSME-VERIFY-KEY.indication.
- 10822 3. The StandardKeyType shall be set to the StandardKeyType of the
10823 APSME-VERIFY-KEY.indication.
- 10824 4. The message shall be APS encrypted only if the Status code is SUCCESS.

10825 **4.4.8 Confirm-Key Services**

10826 **4.4.8.1 APSME-CONFIRM-KEY.request**

10827 This primitive allows a Trust Center to respond to a device requesting to verify its Trust Center Link Key.

10828 **4.4.8.1.1 Semantics of the Service Primitive**

10829 The primitive shall provide the following interface:

10831 APSME-CONFIRM-KEY.request	{
	Status
	DestAddress,
	StandardKeyType
	}

10836 Table 4.25 specifies the parameters of the APSME-CONFIRM-KEY.request primitive.

10837

10838 **Table 4.25 APSME-CONFIRM-KEY.request Parameters**

Parameter Name	Type	Valid Range	Description
Status	Integer	0x00 – 0xFF	A value indicating the success or failure of a previous attempt to verify the trust center link key. See Table 2.27.
DestAddress	Device Address	Any valid 64-bit address	The extended 64-bit address of the device that sent the verify-key command.
StandardKeyType	Octet	0x00-0xFF	Type of key being verified. See Table 4.9.

4.4.8.1.2 When Generated

10840 The Security Manager shall generate this primitive when it wants to respond to a previously received
10841 APSME-VERIFY-KEY.indication.

4.4.8.1.3 Effect on Receipt

10843 On receipt of the APSME-CONFIRM-KEY.request primitive the following shall be performed:

- 10844 1. If the device is not the Trust Center, this is not a valid request. The request shall be dropped and
10845 no further processing shall be done.
- 10846 2. If the StandardKeyType parameter is not equal to 0x04 (Trust Center Link Key), the request is in-
10847 valid. No further processing shall be done.
 - 10848 a. If the apsTrustCenterAddress of the AIB is set to 0xFFFFFFFFFFFFFF, the device is
10849 operating in distributed Trust Center mode and this is not a valid request. The request
10850 shall be dropped and no further processing shall be done.
 - 10851 3. The device shall find the corresponding entry in the *apsDeviceKeyPairSet* attribute of the AIB by
10852 examining the DeviceAddress of all entries and comparing it to the DestAddress of this primitive.
10853 If no match is found, the request is invalid.
 - 10854 a. The device shall send an APS Command Confirm Key Response to the DestAddress set-
10855 ting the StandardKeyType to the StandardKeyType of this primitive, the Status in the
10856 Command to FAILURE. The APS Command shall not be APS encrypted.
 - 10857 b. No further processing shall be done.
 - 10858 4. The device shall send an APS Command Confirm Key Response to the DestAddress setting the
10859 StandardKeyType to the StandardKeyType of this primitive, the Status in the Command to the
10860 Status passed to this primitive. The APS Command shall be APS encrypted.
 - 10861 5. The device shall set the IncomingFrameCounter of the *apsDeviceKeyPairSet* entry to 0.

10863 **4.4.8.2 APSME-CONFIRM-KEY.indication**

10864 This primitive notifies a device of the result of a previous APSME-VERIFY-KEY.request and allows it to
10865 remove a provisional link key used for joining.

10866 **4.4.8.2.1 Semantics of the Service Primitive**

10867 The primitive shall provide the following interface:

10869 APSME-CONFIRM-KEY.indication {
10870 Status
10871 SrcAddress,
10872 StandardKeyType,
10873 }
10874 Table 4.26 specifies the parameters of the APSME-CONFIRM-KEY.indication primitive.
10875

10876 **Table 4.26 APSME-CONFIRM-KEY.indication Parameters**

Parameter Name	Type	Valid Range	Description
Status	Integer	0x00 – 0xFF	The result of the APSME-VERIFY-KEY.request operation.
SrcAddress	Device Address	Any valid 64-bit address	The extended 64-bit address of the device that sent the verify-key command.
StandardKeyType	Octet	0x00-0xFF	Type of key being verified. See Table 4.9.

10877 **4.4.8.2.2 When Generated**

10878 The APSME shall generate this primitive when it receives an APS Command Confirm Key.

10879 **4.4.8.2.3 Effect on Receipt**

10880 On receipt of the APSME-CONFIRM-KEY.indication primitive the following shall be performed:

1. If the message is a NWK broadcast, the request shall be dropped and no further processing shall be done.
2. If the local device is the Trust Center, this primitive is invalid. No further processing shall be done.
3. If the Status parameter is not equal to 0x00 (Success), the operation was unsuccessful. No further processing shall be done.
4. If the StandardKeyType parameter is not equal to 0x04 (Trust Center Link Key), this primitive is invalid. No further processing shall be done.
5. If the apsTrustCenterAddress of the AIB is 0xFFFFFFFFFFFFFF (indicating a distributed security network), this primitive is invalid. No further processing shall be done.

- 10891 6. If the SrcAddress parameter is not the equal to the apsTrustCenterAddress of the AIB, then this
10892 primitive shall be silently dropped. No further processing shall be done.
10893 7. The device shall find the corresponding entry in the apsDeviceKeyPairSet of the AIB where the
10894 DeviceAddress is equal to the apsTrustCenterAddress of the AIB. If no entry can be found, no
10895 further processing shall be done.
10896 8. The device shall set the keyAttributes of the corresponding apsDeviceKeyPairSet entry to 0x02
10897 (VERIFIED_KEY).
10898 9. The device shall set the IncomingFrameCounter of the corresponding apsDeviceKeyPairSet entry
10899 to 0.

10900

10901 **4.4.9 Secured APDU Frame**

10902 The APS layer frame format consists of APS header and APS payload fields (see Figure 4.6). The APS
10903 header consists of frame control and addressing fields. When security is applied to an APDU frame, the
10904 security bit in the APS frame control field shall be set to 1 to indicate the presence of the auxiliary frame
10905 header. The format for the auxiliary frame header is given in section 4.5.1. The format of a secured APS
10906 layer frame is shown in Figure 4.6. The auxiliary frame header is situated between the APS header and
10907 payload fields.²

10908 **Figure 4.6 Secured APS Layer Frame Format**

Octets: Variable	5 or 13	Variable	
Original APS header ([B7], clause 7.1)	Auxiliary frame header	Encrypted payload	Encrypted message integrity code (MIC)
		Secure frame payload = output of CCM	
Full APS header		Secured APS payload	

² Note: Section 4.4.9 is moved text, not added. Moved from section 4.4.6.3

4.4.10 Command Frames

The APS layer command frame formats are given in this clause.

All APS command frames shall set their APS frame control field as follows:

1. Set the frame type sub-field to 0x01 (Command)
2. Set the delivery-mode sub-field to 0x00 (Unicast) or 0x10 (broadcast)
3. Set the ACK format bit to 0.
4. Set the ACK request bit to 0.
5. Set the extended nonce sub-field to 1.
6. Set the security bit according to section 4.4.1.3 Security Processing of APS Commands.

Command identifier values are shown in Table 4.27.

Table 4.27 Command Identifier Values

Command Identifier	Value
Reserved	0x01
Reserved	0x02
Reserved	0x03
Reserved	0x04
APS_CMD_TRANSPORT_KEY	0x05
APS_CMD_UPDATE_DEVICE	0x06
APS_CMD_REMOVE_DEVICE	0x07
APS_CMD_REQUEST_KEY	0x08
APS_CMD_SWITCH_KEY	0x09
Reserved	0x0A
Reserved	0x0B
Reserved	0x0C
Reserved	0x0D
APS_CMD_TUNNEL	0x0E
APS_CMD_VERIFY_KEY	0x0F

Command Identifier	Value
APS_CMD_CONFIRM_KEY	0x10

10920

4.4.10.1 Transport-Key Commands

The transport-key command frame shall be formatted as illustrated in Figure 4.7. The optional fields of the APS header portion of the general APS frame format shall not be present.

Figure 4.7 Transport-Key Command Frame

Octets: 1	1	1	1	Variable
Frame control	APS counter	APS command identifier	Standard-KeyType	Key descriptor
APS header	Payload			

4.4.10.1.1 Command Identifier Field

The command identifier field shall indicate the transport-key APS command type (APS_CMD_TRANSPORT_KEY, seeTable 4.27).

4.4.10.1.2 StandardKeyType Field

This field is 8 -bits in length and describes the type of key being transported. The different types of keys are enumerated in Table 4.9.

4.4.10.1.3 Key Descriptor Field

This field is variable in length and shall contain the actual (unprotected) value of the transported key along with any relevant identification and usage parameters. The information in this field depends on the type of key being transported (as indicated by the StandardKeyType field — seeTable 4.9) and shall be set to one of the formats described in the following subsections.

4.4.10.1.3.1 Trust Center Link Key Descriptor Field

If the key type field is set to 4, the key descriptor field shall be formatted as shown in Figure 4.8.

Figure 4.8 Trust Center Link Key Descriptor Field in Transport-Key Command

Octets: 16	8	8
Key	Destination address	Source address

The key sub-field shall contain the link key that should be used for APS encryption.

The destination address sub-field shall contain the address of the device which should use this link key.

The source address sub-field shall contain the address of the Trust Center that sent the link key.

4.4.10.1.3.2 Network Key Descriptor Field

If the key type field is set to 1 this field shall be formatted as shown in Figure 4.9.

10945

Figure 4.9 Network Key Descriptor Field in Transport-Key Command

Octets: 16	1	8	8
Key	Sequence number	Destination address	Source address

- 10946 The key sub-field shall contain a network key.
 10947 The sequence number sub-field shall contain the sequence number associated with this network key.
 10948 The destination address sub-field shall contain the address of the device which should use this network key.
 10949 If the network key is sent to a broadcast address, the destination address subfield shall be set to the all-zero string and shall be ignored upon reception.
 10950 The source address field sub-field shall contain the address of the device (for example, the Trust Center) which originally sent this network key.
 10951 The source address field shall contain 0xFFFFFFFFFFFFFF in a distributed security network. This indicates to the receiving device this is a distributed security network with no Trust Center.

4.4.10.1.3.3 Application Link Key Descriptor Field

If the key type field is set to 2 or 3, this field shall be formatted as shown in Figure 4.10.

10957

Figure 4.10 Application Link Key Descriptor in Transport-Key Command

Octets: 16	8	1
Key	Partner address	Initiator flag

- 10958 The key sub-field shall contain a link key that is shared with the device identified in the partner address sub-field.
 10959 The partner address sub-field shall contain the address of the other device that was sent this link key.
 10960 The initiator flag sub-field shall be set to 1 if the device receiving this packet requested this key. Otherwise, this sub-field shall be set to 0.

4.4.10.2 Update Device Commands

The APS command frame used for device updates is specified in this clause. The optional fields of the APS header portion of the general APS frame format shall not be present.

The update-device command frame shall be formatted as illustrated in Figure 4.11.

10967

Figure 4.11 Update-Device Command Frame Format

Octets: 1	1	1	8	2	1
Frame control	APS counter	APS command identifier	Device Address	Device short address	Status
APS Header		Payload			

10968

4.4.10.2.1 Command Identifier Field

- 10969 The command identifier field shall indicate the update-device APS command type (APS_CMD_UPDATE_DEVICE, see Table 4.27).
 10970

10971 **4.4.10.2.2 Device Address Field**

10972 The device address field shall be the 64-bit extended address of the device whose status is being updated.

10973 **4.4.10.2.3 Device Short Address Field**

10974 The device short address field shall be the 16-bit network address of the device whose status is being updated.

10976 **4.4.10.2.4 Status Field**

10977 The status field shall be assigned a value as described for the Status parameter in Table 4.14.

10978 **4.4.10.3 Remove Device Commands**

10979 The APS command frame used for removing a device is specified in this clause. The optional fields of the
10980 APS header portion of the general APS frame format shall not be present. The remove-device command
10981 frame shall be formatted as illustrated in Figure 4.12.

10982 **Figure 4.12 Remove-Device Command Frame Format**

Octets: 1	1	1	8
Frame control	APS counter	APS command identifier	Target address
APS Header		Payload	

10983 **4.4.10.3.1 Command Identifier Field**

10984 The command identifier field shall indicate the remove-device APS command type
10985 (APS_CMD_REMOVE_DEVICE, see Table 4.27).

10986 **4.4.10.3.2 Target Address Field**

10987 The target address field shall be the 64-bit extended address of the device that is requested to be removed
10988 from the network.

10989 **4.4.10.4 Request-Key Commands**

10990 The APS command frame used by a device for requesting a key is specified in this clause. The optional
10991 fields of the APS header portion of the general APS frame format shall not be present.

10992 The request-key command frame shall be formatted as illustrated in Figure 4.13.

10993 **Figure 4.13 Request-Key Command Frame Format**

Octets: 1	1	1	1	0/8
Frame control	APS counter	APS command identifier	RequestKeyType	Partner address
APS Header		Payload		

10994 **4.4.10.4.1 Command Identifier Field**

10995 The command identifier field shall indicate the request-key APS command type
10996 (APS_CMD_REQUEST_KEY, see Table 4.27).

10997 **4.4.10.4.2 RequestKeyType Field**
 10998 The key type field shall be set to the key being requested. Note this Key Type is different than the StandardKeyType values used in Table 4.9 for other APS Commands or other APSME primitives. The RequestKeyType field values for the APS Command Request Key are defined in Table 4.19.
 10999
 11000

11001 **4.4.10.4.3 Partner Address Field**

11002 When the RequestKeyType field is 2 (that is, an application key), the partner address field shall contain the extended 64-bit address of the partner device that shall be sent the key. Both the partner device and the device originating the request-key command will be sent the key.
 11003
 11004

11005 When the RequestKeyType field is 4 (that is, a trust center link key), the partner address field will not be present.
 11006

11007 **4.4.10.5 Switch-Key Commands**

11008 The APS command frame used by a device for switching a key is specified in this clause. The optional fields of the APS header portion of the general APS frame format shall not be present.
 11009

11010 The switch-key command frame shall be formatted as illustrated in Figure 4.14.

11011 **Figure 4.14 Switch-key Command Frame Format**

Octets: 1	1	1	1
Frame control	APS counter	APS command identifier	Sequence number
APS Header		Payload	

11012 **4.4.10.5.1 Command Identifier Field**

11013 The command identifier field shall indicate the switch-key APS command type
 11014 (APS_CMD_SWITCH_KEY, see Table 4.27).

11015 **4.4.10.5.2 Sequence Number Field**

11016 The sequence number field shall contain the sequence number identifying the network key to be made active.
 11017

11018 **4.4.10.6 Tunnel Command**

11019 The APS command frame used by a device for sending a command to a device that lacks the current network key is specified in this clause. The optional fields of the APS header portion of the general APS frame format shall not be present. The tunnel-key command frame is sent unsecured.
 11020
 11021

11022 The tunnel-key command frame shall be formatted as illustrated in Figure 4.15.
 11023

Figure 4.15 Tunnel Command Frame Format

Octets:1	1	1	8	2	13	Variable	4
Frame control	APS counter	APS command identifier	Destination address	Tunneled APS header	Tunneled auxiliary frame	Tunneled command	Tunneled APS MIC
APS Header		Payload					

11024 **4.4.10.6.1 Command Identifier Field**

11025 The command identifier field shall indicate the tunnel APS command type (APS_CMD_TUNNEL, see Ta-
11026 ble 4.27).

11027 **4.4.10.6.2 Destination Address**

11028 The destination address field shall be the 64-bit extended address of the device that is to receive the tun-
11029 neled command.

11030 **4.4.10.6.3 Tunneled Auxiliary Frame Field**

11031 The tunneled auxiliary frame field shall be the auxiliary frame (see section 4.5.1) used to encrypt the tun-
11032 neled command. The auxiliary frame shall indicate that a link key was used and shall include the extended
11033 nonce field.

11034 **4.4.10.6.4 Tunneled Command Field**

11035 The tunneled command field shall be the APS command frame to be sent to the destination.

11036

11037 **4.4.10.7 Verify-Key Command**

11038 This APS command is used by a joining device to verify its updated link key with the peer device, such as
11039 the Trust Center.

11040 The Verify-Key Command frame is formatted as illustrated in Figure 4.16.

11041

11042

Octets:1	1	1	1	8	16
Frame control	APS counter	APS command identifier	Standard Key Type	Source address	Initiator Verify-Key Hash Value
APS Header		APS Payload			

11043

11044 **4.4.10.7.1 Command Identifier Field**

11045 The command identifier field shall indicate the verify-key request command type
11046 (APS_CMD_VERIFY_KEY, see Table 4.27).

11047 **4.4.10.7.2 StandardKeyType Field**

11048 This is the type of key being verified. See Table 4.9.

11049 **4.4.10.7.3 Source Address**

11050 This Source address field shall be the 64-bit extended device of the partner device that the destination
11051 shares the link key with.

11052 **4.4.10.7.4 Initiator Verify-Key Hash Value**

11054 This value is the outcome of executing the specialized keyed hash function specified in section B.1.4 using
11055 a key with the 1-octet string ‘0x03’ as the input string. The resulting value shall NOT be used as a key for
11056 encryption or decryption.

11057

11058

11059 **4.4.10.8 Confirm-Key Command**

11060 This APS command is used by a device (such as the trust center) to confirm its updated link key with the
11061 peer device.

11062 The Confirm-Key command frame is formatted as illustrated in Figure 4.17.

11063

11064 **Figure 4.17 Confirm-Key Command Frame**

Octets:1	1	1	1	1	8
Frame control	APS counter	APS command identifier	Status	StandardKeyType	Destination address
APS Payload		APS Payload			

11065

11066 **4.4.10.8.1 Command Identifier Field**

11067 The command identifier field shall indicate the Confirm-Key command type
11068 (APS_CMD_VERIFY_KEY_RESPONSE, see Table 4.27).

11069 **4.4.10.8.2 Status**

11070 This will be the 1-byte status code indicating the result of the operation. See Table 2.27.

11071 **4.4.10.8.3 StandardKeyType**

11072 This is the type of key being verified. See Table 4.9.

11073 **4.4.10.8.4 Destination Address**

11074 This destination address field shall be the 64-bit extended device of the source address of the Verify-Key
11075 message.

11076

11077 **4.4.11 Security-Related AIB Attributes**

11078 The AIB contains attributes that are required to manage security for the APS layer. Each of these attributes
11079 can be read or written using the APSME-GET.request and APSME-SET.request primitives, respectively.
11080 The security-related attributes contained in the APS PIB are presented in Table 4.29.

11081

Table 4.28 AIB Security Attributes

Attribute	Identifier	Type	Range	Description	Default
<i>apsDeviceKeyPairSet</i>	0xaa	Set of key-pair descriptor entries. See Table 4.39.	Variable	A set of key-pair descriptors containing link keys shared with other devices.	-
<i>apsTrustCenterAddress</i>	0xab	Device address	Any valid 64-bit address	Identifies the address of the device's Trust Center. If this value is 0xFFFFFFFFFFFFFF, this means that there is no Trust Center in the network and the network is operating in distributed security mode.	-
<i>apsSecurityTimeOutPeriod</i>	0xac	Integer	0x0000-0xFFFF	The period of time a device will wait for the next expected security protocol frame (in milliseconds).	Defined in stack profile
<i>trustCenterPolicies</i>	0xad	-	Variable	A set of policies encoded in the trust center on how it deals with various security events. See Table 4.32.	

11082

11083

Table 4.29 Elements of the Key-Pair Descriptor

Name	Type	Range	Description	Default
DeviceAddress	Device address	Any valid 64-bit address	Identifies the address of the entity with which this key-pair is shared.	-
KeyAttributes	Enumeration	0x00 – 0x02	This indicates attributes about the key. 0x00 = PROVISIONAL_KEY 0x01 = UNVERIFIED_KEY 0x02 = VERIFIED_KEY	

LinkKey	Set of 16 octets	-	The actual value of the link key.	-
OutgoingFrameCounter	Set of 4 octets	0x00000000-0xFFFFFFFF	Outgoing frame counter for use with this link key.	0x00000000
IncomingFrameCounter	Set of 4 octets	0x00000000-0xFFFFFFFF	Incoming frame counter value corresponding to <i>DeviceAddress</i> .	0x00000000
apsLinkKeyType	Enumeration	0x00 – 0x01	The type of link key in use. This will determine the security policies associated with sending and receiving APS messages. 0x00 = Unique Link Key 0x01 = Global Link Key	0x00

4.5 Common Security Elements

This clause describes security-related features that are used in more than one ZigBee layer. The NWK and APS layers shall use the auxiliary header as specified in section 4.5.1 and the security parameters specified in section 4.5.2. The formatting of all frames and fields in this specification are depicted in the order in which they are transmitted by the NWK layer, from left to right, where the leftmost bit is transmitted first in time. Bits within each field are numbered from 0 (leftmost and least significant) to k-1 (rightmost and most significant), where the length of the field is k bits. Fields that are longer than a single octet are sent to the next layer in the order from the octet containing the lowest numbered bits to the octet containing the highest numbered bits.

4.5.1 Auxiliary Frame Header Format

The auxiliary frame header, as illustrated by Figure 4.18, shall include a security control field and a frame counter field, and may include a sender address field and key sequence number field.

Figure 4.18 Auxiliary Frame Header Format

Octets: 1	4	0/8	0/1
Security control	Frame counter	Source address	Key sequence number

4.5.1.1 Security Control Field

The security control field shall consist of a security level, a key identifier, and an extended nonce sub-field and shall be formatted as shown in Figure 4.19.

11100

11101

Figure 4.19 Security Control Field Format

Bit: 0-2	3-4	5	6-7
Security level	Key identifier	Extended nonce	Reserved

11102

4.5.1.1.1 Security Level Sub-Field

11103
11104
11105
11106
11107
11108
11109

The security level identifier indicates how an outgoing frame is to be secured, how an incoming frame purportedly has been secured; it also indicates whether or not the payload is encrypted and to what extent data authenticity over the frame is provided, as reflected by the length of the message integrity code (MIC). The bit-length of the MIC may take the values 0, 32, 64 or 128 and determines the probability that a random guess of the MIC would be correct. The security properties of the security levels are listed in Table 4.29. Note that security level identifiers are not indicative of the relative strength of the various security levels. Also note that security levels 0 and 4 should not be used for frame security.

11110

11111

Table 4.30 Security Levels Available to the NWK, and APS Layers

Security Level Identifier	Security Level Sub-Field	Security Attributes	Data Encryption	Frame Integrity (length M of MIC, in Number of Octets)
0x00	'000'	None	OFF	NO (M = 0)
0x01	'001'	MIC-32	OFF	YES (M=4)
0x02	'010'	MIC-64	OFF	YES (M=8)
0x03	'011'	MIC-128	OFF	YES (M=16)
0x04	'100'	ENC	ON	NO (M = 0)
0x05	'101'	ENC-MIC-32	ON	YES (M=4)
0x06	'110'	ENC-MIC-64	ON	YES (M=8)
0x07	'111'	ENC-MIC-128	ON	YES (M=16)

11112

4.5.1.1.2 Key Identifier Sub-Field

11113
11114
11115

The key identifier sub-field consists of two bits that are used to identify the key used to protect the frame. The encoding for the key identifier sub-field shall be as listed in Table 4.30. Key derivation is described in section 4.5.3.

11116

11117

Table 4.31 Encoding of the Key Identifier Sub-Field

Key Identifier	Key Identifier Sub-Field (Figure 4.19)	Description
0x00	'00'	A data key.
0x01	'01'	A network key.
0x02	'10'	A key-transport key.
0x03	'11'	A key-load key.

11118

4.5.1.1.3 Extended Nonce Sub-Field

11119
11120

The extended nonce sub-field shall be set to 1 if the sender address field of the auxiliary header is present. Otherwise, it shall be set to 0.

11121

4.5.1.2 Counter Field

11122

The counter field is used to provide frame freshness and to prevent processing of duplicate frames.

11123

4.5.1.3 Source Address Field

11124
11125
11126

The source address field shall only be present when the extended nonce sub-field of the security control field is 1. When present, the source address field shall indicate the extended 64-bit address of the device responsible for securing the frame.

11127

4.5.1.4 Key Sequence Number Field

11128
11129
11130

The key sequence number field shall only be present when the key identifier subfield of the security control field is 1 (that is, a network key). When present, the key sequence number field shall indicate the key sequence number of the network key used to secure the frame.

11131

4.5.2 Security Parameters

11132

This section specifies the parameters used for the CCM security operations.

11133

4.5.2.1 CCM Mode of Operation and Parameters

11134
11135

Applying security to a NWK or APS frame on a particular security level corresponds to a particular instantiation of the AES-CCM mode of operation as specified in section B.1.2.

11136

The nonce shall be formatted as specified in section 4.5.2.2.

11137
11138
11139

Table 4.29 gives the relationship between the security level sub-field of the security control field (Figure 4.19), the security level identifier, and the CCM encryption/authentication properties used for these operations.

11140

4.5.2.2 CCM Nonce

11141
11142
11143
11144
11145
11146
11147
11148
11149
11150

The nonce input used for the CCM encryption and authentication transformation and for the CCM decryption and authentication checking transformation consists of data explicitly included in the frame and data that both devices can independently obtain. Figure 4.20 specifies the order and length of the subfields of the CCM nonce. The nonce's security control and frame counter fields shall be the same as the auxiliary header's security control and frame counter fields (as defined in section 4.5.1) of the frame being processed. The nonce's source address field shall be set to the extended 64-bit IEEE address of the device originating security protection of the frame. When the extended nonce sub-field of the auxiliary header's security control field is 1, the extended 64-bit IEEE address of the device originating security protection of the frame shall correspond to the auxiliary header's source address field (as defined in section 4.5.1) of the frame being processed.

11151

Figure 4.20 CCMNonce

Octets: 8	4	1
Source address	Frame counter	Security control

11152

4.5.3 Cryptographic Key Hierarchy

11153
11154

The link key established between two (or more) devices is used to determine related secret keys, including data keys, key-transport keys, and key-load keys. These keys are determined as follows:

11155
11156
11157
11158
11159

1. *Key-Transport Key*. This key is the outcome of executing the specialized keyed hash function specified in section B.1.4 under the link key with the 1-octet string '0x00' as the input string.
2. *Key-Load Key*. This key is the outcome of executing the specialized keyed hash function specified in section B.1.4 under the link key with as input string the 1-octet string '0x02' as the input string.
3. *Data Key*. This key is equal to the link key.

11160
11161

All keys derived from the link key shall share the associated frame counters. Also, all layers of ZigBee shall share the active network key and associated outgoing and incoming frame counters.

11162

4.5.4 Implementation Requirements

11163

This clause provides requirements that should be followed to ensure a secure implementation.

11164

4.5.4.1 Random Number Generator

11165
11166
11167

A ZigBee device generating random keys for distribution requires a strong method of random number generation. For example, when link keys are pre-installed (for example, in the factory), a random number may not be needed.

11168
11169
11170
11171

In all cases that require random numbers, it is critical that the random numbers are not predictable or have enough entropy, so an attacker will not be able determine them by exhaustive search. Random number generation shall meet the random number tests specified in FIPS140- 2 [B13]. Methods for generation of random numbers include:

11172
11173
11174
11175

1. Base the random number on random clocks and counters within the ZigBee hardware;
2. Base the random number on random external events;
3. Seed each ZigBee device with a good random number from an external source during production. This random number can then be used as a seed to generate additional random numbers.

11176 A combination of these methods can be used. Since the random number generation is likely integrated into
11177 the ZigBee IC, its design — and hence the ultimate viability of any encryption/security scheme — is left up
11178 to the IC manufacturers.

4.6 Functional Description

11180 This section provides detailed descriptions of how the security services shall be used in a ZigBee network.
11181 A description of the security initialization responsibilities for a device starting a network is given in section
11182 4.6.1. A brief description of the Trust Center application is given in section 4.6.2. Detailed security proce-
11183 dures are given in section 4.6.3.

4.6.1 ZigBee Security Initialization

11185 The device starting a network shall configure the security level of the network by setting the *nwkSecurityLevel* attribute in the NIB. If the *nwkSecurityLevel* attribute is set to zero, the network will be unse-
11186 cured, otherwise it will be secured.
11187

11188 The *key* value of the *nwkSecurityMaterialSet* attribute shall be set to any non-zero, random number within
11189 the range of all possible values. See section 4.5.4.1 for the requirements of random number generation.
11190 The *KeySeqNumber* of the *nwkSecurityMaterialSet* shall be set to 0.

11191 If it is a centralized security network then the device shall configure the address of the Trust Center by set-
11192 ting the AIB attribute *apsTrustCenterAddress*. The device forming the network may also set the *apsTrustCenterAddress*
11193 to 0xFFFFFFFFFFFFFFFFF indicating a distributed security network.

4.6.2 Trust Center Application

11195 The Trust Center application runs on a device trusted by devices within a ZigBee network to distribute keys
11196 for the purpose of network and end-to-end application configuration management. The Trust Center shall
11197 configure network security policies and shall be used to help establish end-to-end application keys. These
11198 keys shall be generated at random unless a key establishment protocol is used.

4.6.2.1 Distributed Security Mode

11200 In Distributed Security Mode, there is no unique Trust Center in the network. Keys are distributed to
11201 joining devices by routers in the network using the standard transport key commands, or by other out of
11202 band methods.

4.6.2.2 Centralized Security Mode

11204 The centralized security mode of the Trust Center is designed for applications where a centralized security
11205 device and set of security policies is required. In this mode, the Trust Center may maintain a list of devices,
11206 link keys and network keys with all the devices in the network; however, it shall maintain a network key
11207 and controls policies of network admittance. In this mode, the *nwkAllFresh* attribute in the NIB shall be set
11208 to FALSE.

11209 Each device that joins the network securely shall either have a Global Link key or a unique link key de-
11210 pending upon the application in use. It is required that the trust center have prior knowledge of the value of
11211 the link key and the type (Global or unique) in order to securely join the device to the network. A Global
11212 Link key has the advantage that the memory required by the Trust Center does not grow with the number of
11213 devices in the network. A unique link key has the advantage of being unique for each device on the net-
11214 work and application communications can be secured from other devices on the network. Both types of
11215 keys may be used on the network, but a device shall only have one type in use per device-key pair.

11216 The security policy settings for centralized security are further detailed in section 4.7.1.

11217

4.6.3 Security Procedures

11218
11219
11220

This section gives message sequence charts for joining a secured network, authenticating a newly joined device, updating the network key, recovering the network key, establishing end-to-end application keys, and leaving a secured network.

11221

4.6.3.1 Joining a Secured Network

11222
11223
11224
11225

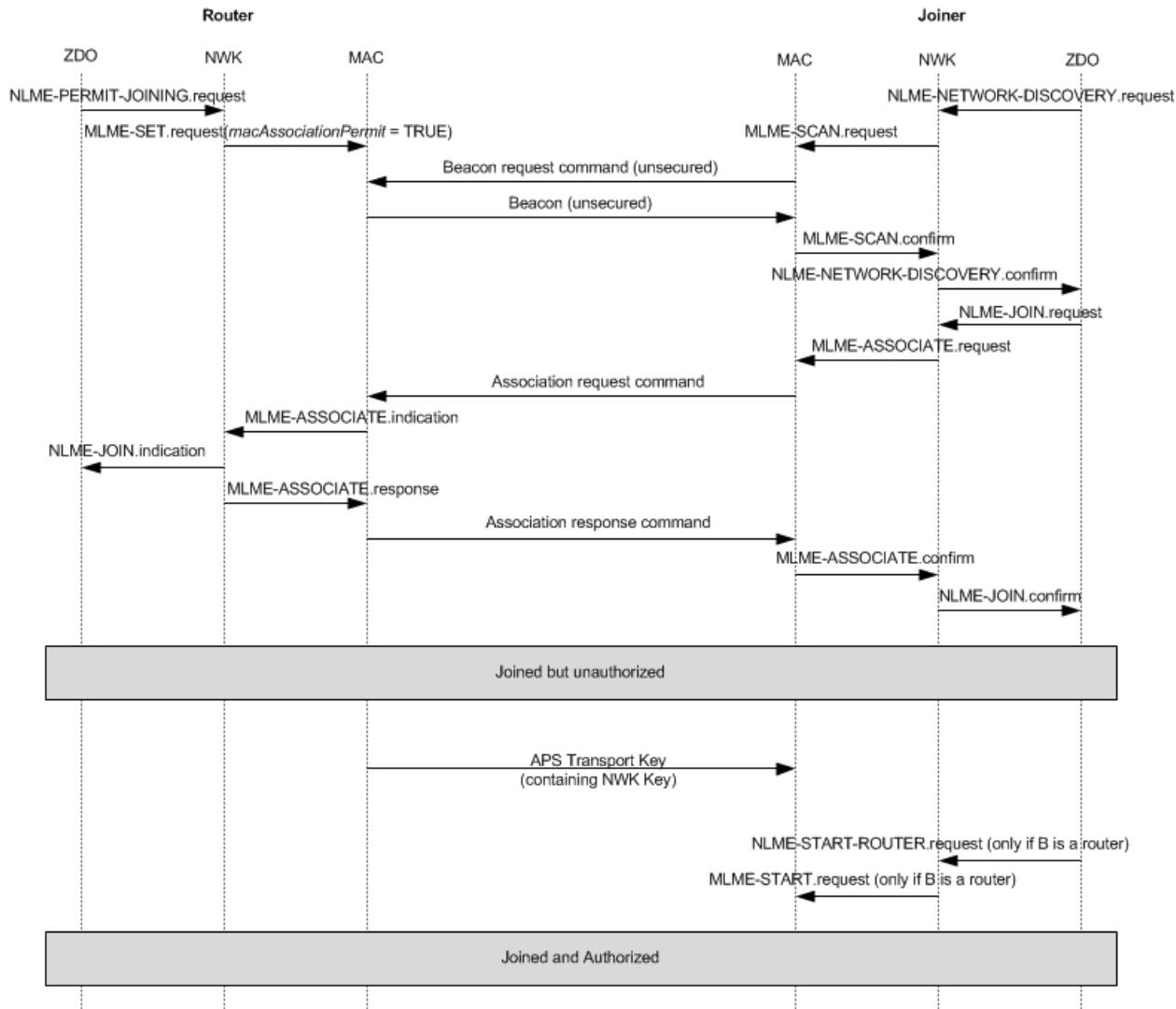
When a device prepares to join a secured network it shall set the AIB attribute *apsLinkKeyType* for its link key with the trust center according to the kind of key it has. If it is using the default trust center link key, or another Global Link key, it shall set *apsLinkKeyType* to 0x01. If it is using a unique link key it shall set *apsLinkKeyType* to 0x00.

11226
11227
11228

Figure 4.21 shows an example message sequence chart ensuing from when a joiner device communicates with a router device to join a secured network. A device that is operating in a network and has missed a network key update may also use these procedures to receive the latest network key.

11229

Figure 4.21 Example of Joining a Secured Network



11230

11231 The joiner device may begin the join procedure by issuing an NLME-NETWORK- DISCOVERY.request
11232 primitive. This primitive will invoke an MLME-SCAN.request primitive which may cause the transmission
11233 of an unsecured beacon request frame (depending on whether the scan is an active or passive scan).

11234 The joiner device receives beacons from nearby routers and the NWK layer issues an
11235 NLME-NETWORK-DISCOVERY.confirm primitive. The NetworkList parameter of this primitive will
11236 indicate all of the nearby PANs. In Figure 4.21, the shown router device has already been placed in a state
11237 such that its beacons have the “association permit” sub-field set to “1” (permit association).

11238 The joiner device shall decide which PAN to join and shall issue the NLME-JOIN.request primitive to join
11239 that PAN. If the joiner already has a network key for this PAN, the SecurityEnable parameter for the
11240 NLME-JOIN.request primitive shall be set to TRUE; otherwise it shall be set to FALSE. As shown in Fig-
11241 ure 4.26, the NLME-JOIN.request primitive causes an association request or rejoin request command to be
11242 sent to the router.

11243 Upon receipt of an association request MAC command, the router shall issue an
11244 MLME-ASSOCIATE.indication primitive. Next, the NWK layer will issue an NLME-JOIN.indication
11245 primitive to the router’s ZDO. The router shall now know the joiner device’s address and security capabili-
11246 ties. The router will also issue an MLME-ASSOCIATE.response. This primitive will cause an association
11247 response command to be sent to the joiner.

11248 Alternatively, upon receipt of a rejoin request network command, the NWK layer will issue an
11249 NLME-JOIN.indication primitive to the router’s ZDO. The router shall now know the joiner device’s address,
11250 security capabilities, and whether the network key was used to secure the rejoin request command.
11251 The router will also issue a rejoin response command to the joiner.

11252 Upon receipt of the association response MAC command or the rejoin response network command, the
11253 joiner shall issue the NLME-JOIN.confirm primitive. The joiner is now declared “joined, but unauthorized”
11254 to the network. The authorization routine (see section 4.6.3.2) shall follow.

11255 If the joiner is not a router, it is declared “joined and authorized” immediately following the successful
11256 completion of the authorization routine.

11257 If the joiner is a router, it is declared “joined and authorization” only after the successful completion of the
11258 authorization routine followed by the initiation of routing operations. Routing operations shall be initiated
11259 by the joiner’s ZDO issuing the NLME-START.request primitive to cause the MLME-START.request
11260 primitive to be sent to the MAC layer of the joiner.

11261 If the router refuses the joiner, its association response frame or rejoin response frame shall contain the as-
11262 sociation status field set to a value other than 0x00, and, after this parameter reaches the ZDO of the joiner
11263 in the NLME-JOIN.confirm primitive, the joiner shall not begin the authorization routine.

11264 **4.6.3.2 Authorization**

11265 Once a device joins a secured network and is declared “joined but unauthorized”, it must be authorized by
11266 receiving an APS transport key command containing the active network key.

11267 **4.6.3.2.1 Router Operation**

11268 If the *apsTrustCenterAddress* is all 0xFFFFFFFFFFFFFF, this indicates a Distributed Security network.
11269 If the *apsTrustCenterAddress* is any other value, it indicates a Centralized Security network.

11270 In centralized security networks, if the router is not the Trust Center, it shall begin the authorization proce-
11271 dure immediately after receipt of the NLME-JOIN.indication primitive by issuing an
11272 APSME-UPDATE-DEVICE.request primitive with the DestAddress parameter set to the *apsTrustCen-*
11273 *terAddress* in the AIB and the DeviceAddress parameter set to the address of the newly joined device. The
11274 Status parameter of this primitive shall be set by the NLME-JOIN.indication primitive parameters accord-
11275 ing to Table 4.31.

11276 In a Distributed Security Network no Update Device message is generated. The router shall issue an
 11277 APSME-TRANSPORT-KEY.request with the StandardKeyType set to 0x01 (Standard Network Key) and
 11278 the key value from the nwkSecurityMaterialSet of the NIB with a KeySeqNumber equal to the nwkAc-
 11279 tiveSeqNumber of the NIB. The message shall be APS encrypted with the Distributed Security Global
 11280 Key in the apsDeviceKeyPairSet.

11281 **Table 4.32 Mapping of NLME-JOIN.indication Parameters to Update Device Status**

NLME-JOIN.indication Parameters			Update Device Status	
Capability Information Bit 6	Method (RejoinNetwork parameter)	Request Secured	Status	Description
0	NWK Rejoin (0x02)	TRUE	0x00	Standard security device secured rejoin
0	MAC Association (0x00)	FALSE	0x01	Standard security device unsecured join
0	NWK Rejoin (0x02)	FALSE	0x03	Standard security device unsecured rejoin

11282
 11283 If the router is the Trust Center, it shall begin the authorization procedure by simply operating as a Trust
 11284 Center.

11285 The router shall not forward messages to a child device, or respond to ZDO requests or NWK command
 11286 requests on that child's behalf, while the value of the relationship field entry in the corresponding
 11287 *nwkNeighborTable* in the NIB is 0x05 (unauthorized child).

11288 **4.6.3.2.2 Trust Center Operation**

11289 The Trust Center role in the authorization procedure shall be activated upon receipt of an incoming up-
 11290 date-device command or immediately after receipt of the NLME-JOIN.indication primitive (in the case
 11291 where the router is the Trust Center). The Trust Center behaves differently depending on the following
 11292 factors:

11293 Whether the Trust Center decides to allow any device to perform a first time join (for example, the Trust
 11294 Center is in a mode that allows new devices to join).

11295 If the Trust Center Policies require prior knowledge of the device to allow joining

11296 If, at any time during the authorization procedure, the Trust Center decides not to allow the new device to
 11297 join the network (for example, a policy decision or a failed higher level key-establishment protocol), it shall
 11298 take actions to remove the device from the network. If the Trust Center is not the router of the newly joined
 11299 device, it may remove the device from the network by issuing the APSME-REMOVE-DEVICE. request
 11300 primitive with the ParentAddress parameter set to the address of the router originating the update-device
 11301 command and the ChildAddress parameter set to the address of the joined (but unauthorized) device. Al-
 11302 ternatively the Trust Center may let an unauthorized device just timeout; in that case the Trust Center will
 11303 not send a removal message.

11304 **4.6.3.2.2.1 Initial Key Distribution**

11305 After being activated for the authorization procedure, the Trust Center shall determine whether or not to
11306 allow the device onto the network. This decision will be based on its own security policies, see section
11307 4.7.1. If it decides to allow the device onto the network, it shall send the device the active network key by
11308 issuing the APSME-TRANSPORT-KEY.request primitive with the DestAddress parameter set to the ad-
11309 dress of the newly joined device, and the StandardKeyType parameter set to 0x01 (that is, standard network
11310 key).

11311 The KeySeqNumber sub-parameter of the APSME-TRANSPORT-KEY.request shall be set to the sequence
11312 count value for the active network key and the NetworkKey sub-parameter shall be set to the active net-
11313 work key. The UseParent sub-parameter shall be set to FALSE if the Trust Center is the router; otherwise,
11314 the UseParent sub-parameter shall be set to TRUE and the ParentAddress sub-parameter shall be set to the
11315 address of the router originating the update-device command.

11316 **4.6.3.2.3 Joining Device Operation**

11317 After successfully joining or rejoining a secured network, the joining device shall set the *nwkSecurityLevel*
11318 attribute in the NIB to the values indicated by the stack profile.

11319 A joined and authorized device shall always apply NWK layer security to outgoing frames unless the frame
11320 is destined for a newly joined but unauthorized child.

11321 In a secured network, if the device does not become authorized within a preconfigured amount of time, it
11322 shall leave the network (see section 4.6.3.6.3).

11323 **4.6.3.2.3.1 Preconfigured Trust Center Link Key**

11324 The joining device shall be preconfigured with a Trust Center link key and wait to receive an active net-
11325 work key encrypted with its preconfigured link key. Upon receipt of the
11326 APSME-TRANSPORT-KEY.indication primitive with the StandardKeyType parameter set to 0x01 (that
11327 is, the standard network key), the joining device shall set the *apsTrustCenterAddress* attribute in its AIB to
11328 the SrcAddress parameter of the APSME-TRANSPORT-KEY.indication primitive. The joining device is
11329 now considered authorized and shall enter the normal operating state for standard security mode.

11330 If the *apsTrustCenterAddress* is set to 0xFFFFFFFFFFFFFF the network is in distributed security mode.
11331 The device shall enter the normal operating state.

11332 Additional application layer security authentication or initialization may be required by the higher layer
11333 specification.

11334 If the joining device did not receive the key via the APSME-TRANSPORT-KEY.indication within the
11335 *apsSecurityTimeOutPeriod* since receiving the NLME-JOIN.confirm primitive, it shall reset and may
11336 choose to start the joining procedure again.

11337

11338

11339 **4.6.3.2.4 Message Sequence Charts**

11340 Figure 4.22 shows the message sequence charts for the authorization procedure when the joining is not to
11341 the Trust Center directly, but through an intermediate router.

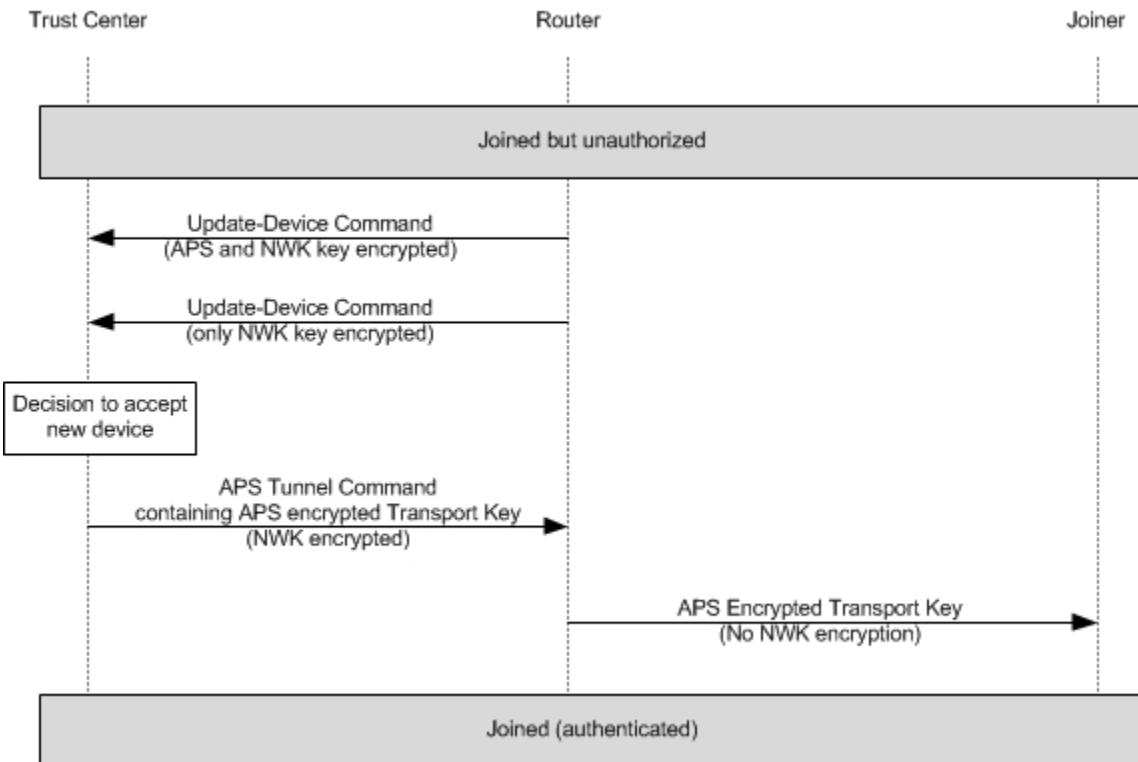
11342 The update-device and tunnel commands communicated between the Trust Center and the router shall be
11343 secured at the NWK layer by the active network key. The transport-key command sent from the router to
11344 the joiner shall not be secured at the network layer. Two copies of the update-device APS command shall
11345 be generated by the intermediary router if the *apsDeviceKeyPairSet* entry for the TC indicates the
11346 *apsLinkKeyType* is 0x01 (Global). One copy shall be encrypted at both the APS and the NWK layer, while
11347 the other copy shall only be encrypted at the NWK layer. This is done due to an interoperability issue
11348 where previously certified Trust Centers may have requirements on the encryption that it accepts for the
11349 update device message.

11350 A device with `apsDeviceKeyPairSet` that has an `apsLinkKeyType` of 0x00 (Unique Link Key) does not
11351 have to generate two update device messages.

11352

11353

Figure 4.22 - Multi-hop Join and Trust Center Rejoin Diagram



11354

11355

4.6.3.3 Rejoining Security

11357 Devices shall follow the procedures described in this section as necessary to support rejoining, in conjunction
11358 with the mechanism described in section 2.5.4.5.2.2.

11359 A device does not have to verify its trust center link key with the APSME-VERIFY-KEY services after a
11360 rejoin.

4.6.3.3.1 Secure Rejoin

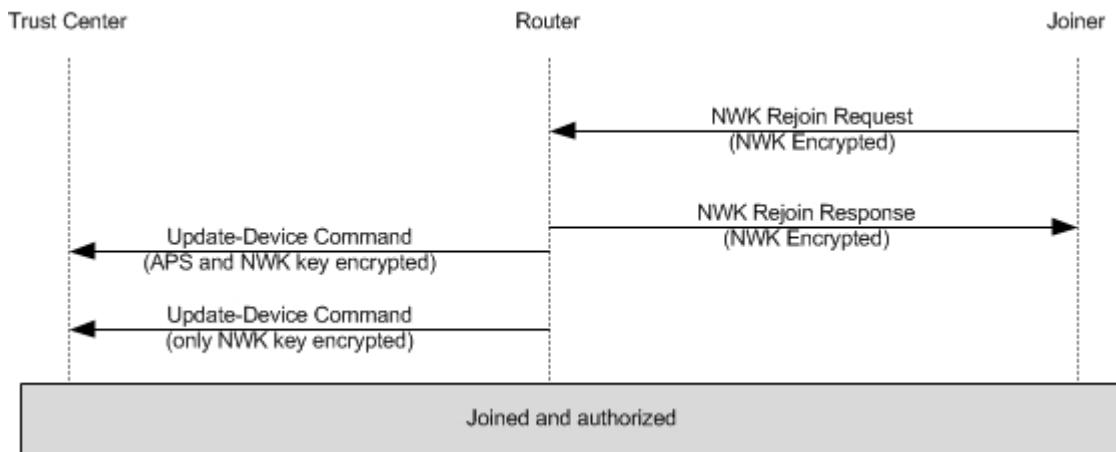
11362 When a device is rejoining and secures the NWK rejoin request command with the active network key, no
11363 further authorization is required beyond validation of the NWK security. Both centralized and distributed
11364 networks may use Secure Rejoin.

11365 Figure 4.23 shows the flow of messages during a secure rejoin. Note that in Distributed network security
11366 the APS Command Update Device shall not be sent.

11367

11368

Figure 4.23 - Secure Rejoin



11369

11370

11371

4.6.3.3.2 Trust Center Rejoin

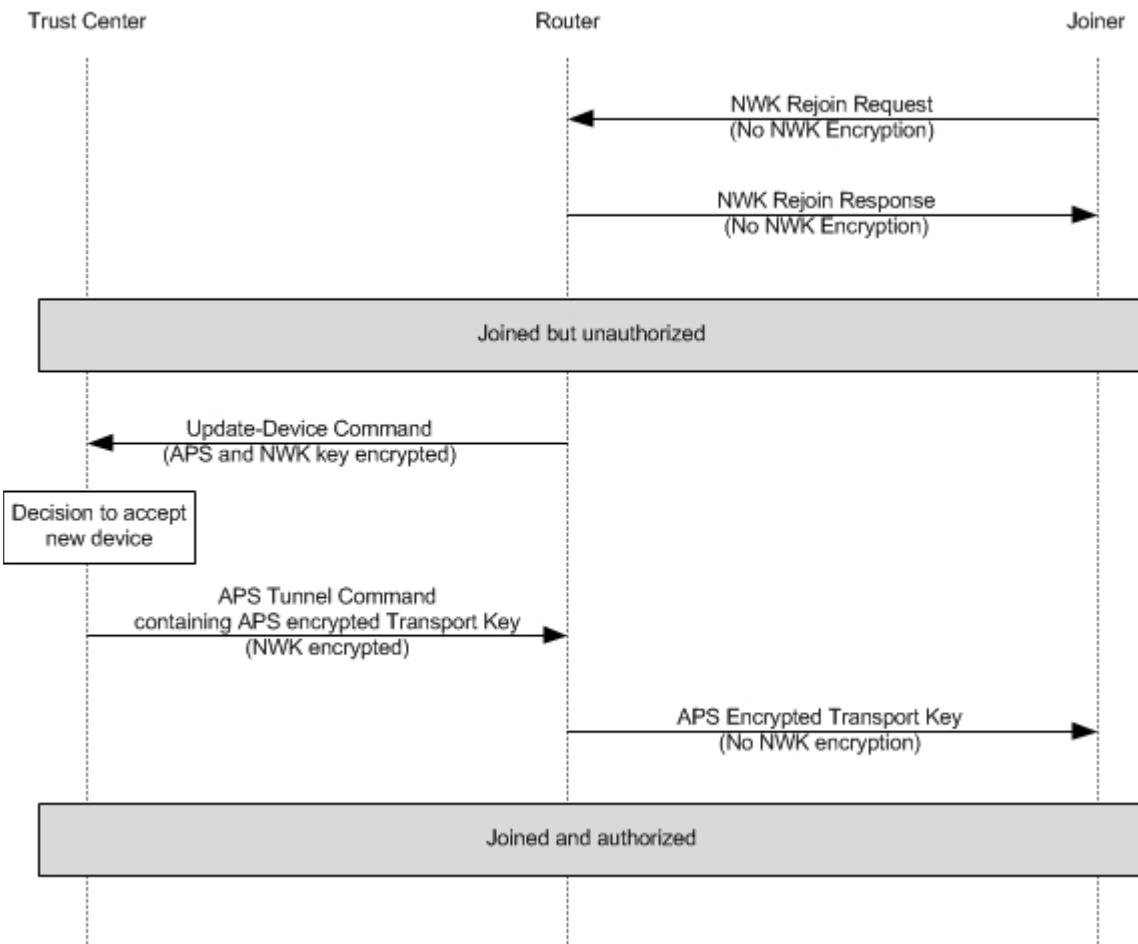
A Trust Center Rejoin is used when a device may no longer have the current network key and therefore should not secure the NWK rejoin command. If the network is using a different network key then the device using the old network key will be rejected. A Trust Center rejoin is a NWK Rejoin command where the command is sent without NWK layer security and allows a device to request the current active network key.

11377

Figure 4.24 illustrates a trust center rejoin operation.

11378

Figure 4.24 - Trust Center Rejoin



11379

11380 A Trust Center Rejoin shall only be allowed in a centralized security network. Attempts to use a Trust
11381 Center rejoin in a distributed security network shall be rejected.

11382

The following sections describe the behavior of the devices in the network and the orphaned devices.

11383

4.6.3.3.3 Coordinator and Router Operation

11384 This text describes the security operations for support of rejoining which are to be carried out by the
11385 ZigBee coordinator and by ZigBee routers that are already operating on the network. These devices will
11386 receive rejoin requests by orphaned devices and will act as follows.

11387 Following the steps described in section 2.5.4.5.2.2, an orphaned device (router or end device) shall be pro-
11388 visionally accepted onto the network by the coordinator or router for at least *apsSecurityTimeOutPeriod*
11389 milliseconds. During this period it shall be required to send at least one correctly formed ZigBee message
11390 secured with the network key to the new parent. If this message successfully passes all the security pro-
11391 cessing steps described in this document, it shall be accepted as a member of the network.

11392 This specification neither specifies nor requires any action from the router or coordinator in the case that a
11393 message from an orphaned device fails security processing above that required by text elsewhere in this
11394 document.

11395 **4.6.3.3.4 Rejoining Device Operation**

11396 Following the steps described in section 2.5.4.5.2.2, an orphaned device (router or end device) shall be pro-
11397 visionally accepted onto the network for at least *apsSecurityTimeOutPeriod* milliseconds. During this pe-
11398 riod, it shall be required to send at least one ZigBee message, secured with the network key to the new
11399 parent.

11400 As normal, the device shall not accept an unsecured network key (having no NWK security) from the Trust
11401 Center.

11402 Note that a ZigBee device may also carry out an orphan scan as described in section 2.5.5.2.2. In this case
11403 it shall, at this time, also follow the steps described in this sub-section.

11404 **4.6.3.4 Network Key Update**

11405 The Trust Center and network devices shall follow the procedures described in this section when updating
11406 the active network key. Updating of the network key is not possible when operating in distributed securi-
11407 ty mode.

11408 **4.6.3.4.1 Trust Center Operation**

11409 When updating a standard network key with a new key of the same type, the Trust Center may broadcast or
11410 unicast the key update. If it chooses to broadcast the new key to all devices on the network it issues the
11411 APSME-TRANSPORT-KEY.request primitive with the DestAddress parameter set to the broadcast ad-
11412 dress and the StandardKeyType parameter set to 0x01 (that is, a network key).

11413 For a unicast key update the Trust Center shall issue multiple APSME-TRANSPORT-KEY.request primi-
11414 tive with the DestAddress set to each device it wants to notify of the new key.

11415 The TransportKeyData sub-parameters shall be set as follows for both unicast and broadcast key updates:

- 11416 • The KeySeqNumber sub-parameter shall be set to the sequence count value for the new network
11417 key.
- 11418 • The NetworkKey sub-parameter shall be set to the new network key.
- 11419 • The UseParent sub-parameter shall be set to FALSE.

11420 If the sequence count for the previously distributed network key is represented as N , then the sequence
11421 count for this new network key shall be $(N+1) \bmod 256$.

11422 The Trust Center may cause a switch to this new key by issuing the APSME-SWITCH-KEY.request primi-
11423 tive with the DestAddress parameter set to the broadcast address and the KeySeqNumber parameter set to
11424 the sequence count value for the updated network key. The switch key shall not be unicast. It shall be
11425 encrypted at the network layer with the current network key.

11426 In centralized security mode, the Trust Center may maintain a list of all of the devices in the network. To
11427 update the active network key using this list, the Trust Center may first send the new network key to each
11428 device on this list and then ask the network to switch to the new key.

11429 **4.6.3.4.2 Network Device Operation**

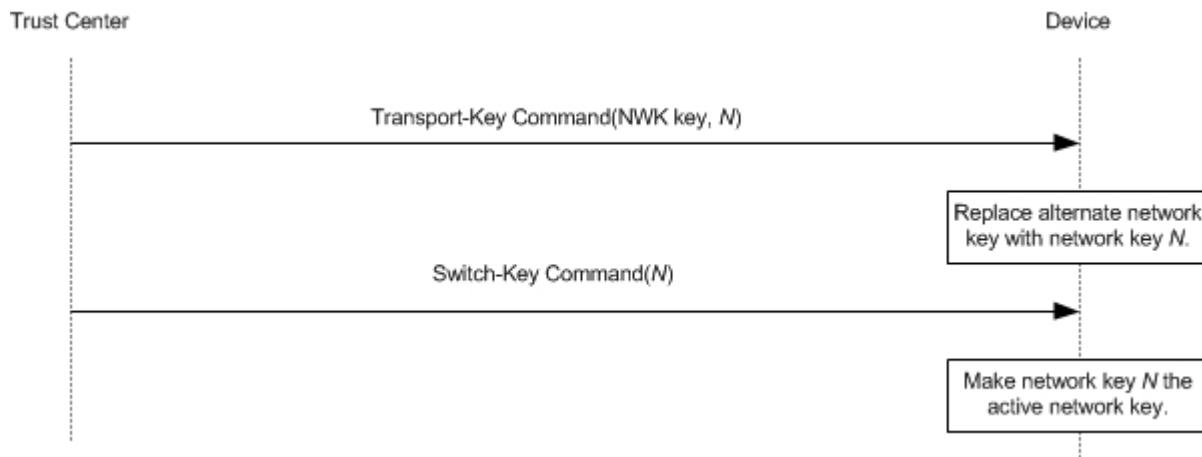
11430 Devices shall be capable of storing 2 network keys, the current and an alternate.

11431 When in the normal operating state and upon receipt of a APSME-TRANSPORT-KEY.indication primitive
11432 with the StandardKeyType parameter set to 0x01 (that is, a network key), a device shall accept the
11433 TransportKeyData parameters as a network key only if the SrcAddress parameter is the same as the Trust
11434 Center's address (as maintained in the *apsTrustCenterAddress* attribute of the AIB). If accepted, the key
11435 and sequence number data contained in the TransportKeyData parameter shall replace the alternate network
11436 key. Otherwise, the key and sequence number data contained in the TransportKeyData parameter shall re-
11437 place the active network key. In either case, all incoming frame counters and the outgoing frame counter of
11438 the appropriate network key shall be set to 0.

11439 When in the normal operating state and upon receipt of an APSME-SWITCH-KEY.indication primitive, a
11440 device shall switch its active network key to the one designated by the KeySeqNumber parameter only if
11441 the SrcAddress parameter is the same as the Trust Center's address (as maintained in the *apsTrustCenterAddress* attribute of the AIB). Figure 4.25 illustrates the procedure.
11442

11443

Figure 4.25 Example Network Key-Update Procedure



11445

4.6.3.4.3 Message Sequence Chart

An example of a successful network key-update procedure for two devices is shown in Figure 4.25. In this example, the Trust Center sends a network key with sequence number N to the device. All devices are required to be capable of storing two network keys, an active and alternate. Upon receipt of the transport-key command, the device replaces its alternate network key with the new network key. Next, upon receipt of the switch-key command, the device makes the new network key the active network key.

11452

4.6.3.5 End-to-End Application Key Establishment

11453 An initiator device, a Trust Center, and a responder device shall follow the procedures described in this
11454 section when establishing a link key for purposes of end-to-end application security between initiator and
11455 responder devices.

11456

4.6.3.5.1 Device Operation

11457 The initiator device shall begin the procedure to establish a link key with a responder device by issuing the
11458 APSME-REQUEST-KEY.request primitive. The DestAddress parameter shall be set to the address of its
11459 Trust Center, the RequestKeyType parameter shall be set to 0x02 (that is, application link key), and the
11460 PartnerAddress parameter shall be set to the address of the responder device.

11461 In a distributed security network where there is not a trust center to authorize the distribution of application
11462 link keys, an initiator device may issue an APSME-TRANSPORT-KEY.request to a responder device
11463 based on application policies on the device.

11464 **4.6.3.5.1.1 Upon Receipt of a Link Key**

11465 Upon receipt of an APSME-TRANSPORT-KEY.indication primitive with the StandardKeyType parameter
11466 set to 0x03 (that is, application link key), a device may accept the TransportKeyData parameters as a link
11467 key with the device indicated by the PartnerAddress parameter only if the SrcAddress parameter is the
11468 same as the *apsTrustCenterAddress* attribute of the AIB. If accepted, the *apsDeviceKeyPairSet* attribute in
11469 AIB table will be updated. A key-pair descriptor in the AIB shall be created (or updated if already present)
11470 for the device indicated by the PartnerAddress parameter, by setting the DeviceAddress element to the
11471 PartnerAddress parameter, the LinkKey element to the link key from the TransportKeyData parameter, and
11472 the OutgoingFrameCounter and IncomingFrameCounter elements to 0 unless the value is the same as the
11473 previous link key.

11474 In the case of a distributed security network, a device may accept an
11475 APSME-TRANSPORT-KEY.indication primitive with the StandardKeyType parameter set to 0x03 (that
11476 is, application link key) from a partner device since no trust center exists. The device and this partner can
11477 then establish an application link key based on the application level policies of the device.

11478

4.6.3.5.2 Trust Center Operation

11480 Upon receipt of APSME-REQUEST-KEY.indication primitives with the StandardKeyType parameter set
11481 to 0x02 (that is, application link key).

11482 The Trust Center shall issue two APSME-TRANSPORT-KEY.request primitives with the StandardKeyType
11483 parameter shall be set to 0x03 (that is, application link key). The first primitive shall have the
11484 DestAddress parameter set to the address of the device requesting the key. The TransportKeyData
11485 sub-parameters shall be set as follows:

- The PartnerAddress sub-parameter shall be set to the PartnerAddress sub-parameter of the
 APSME-REQUEST-KEY.indication primitive's TransportKeyData parameter.
- The Initiator sub-parameter shall be set to TRUE.
- The Key sub-parameter shall be set to a new key K (link key).

11490 The key shall have been generated in a random fashion. The second primitive shall have the DestAddress
11491 parameter set to the PartnerAddress sub-parameter of the APSME-REQUEST-KEY.indication primitive's
11492 TransportKeyData parameter. The TransportKeyData sub-parameters shall be set as follows:

- The PartnerAddress sub-parameter shall be set to the address of the device requesting the key.
- The Initiator sub-parameter shall be set to FALSE.
- The Key sub-parameter shall be set to K.

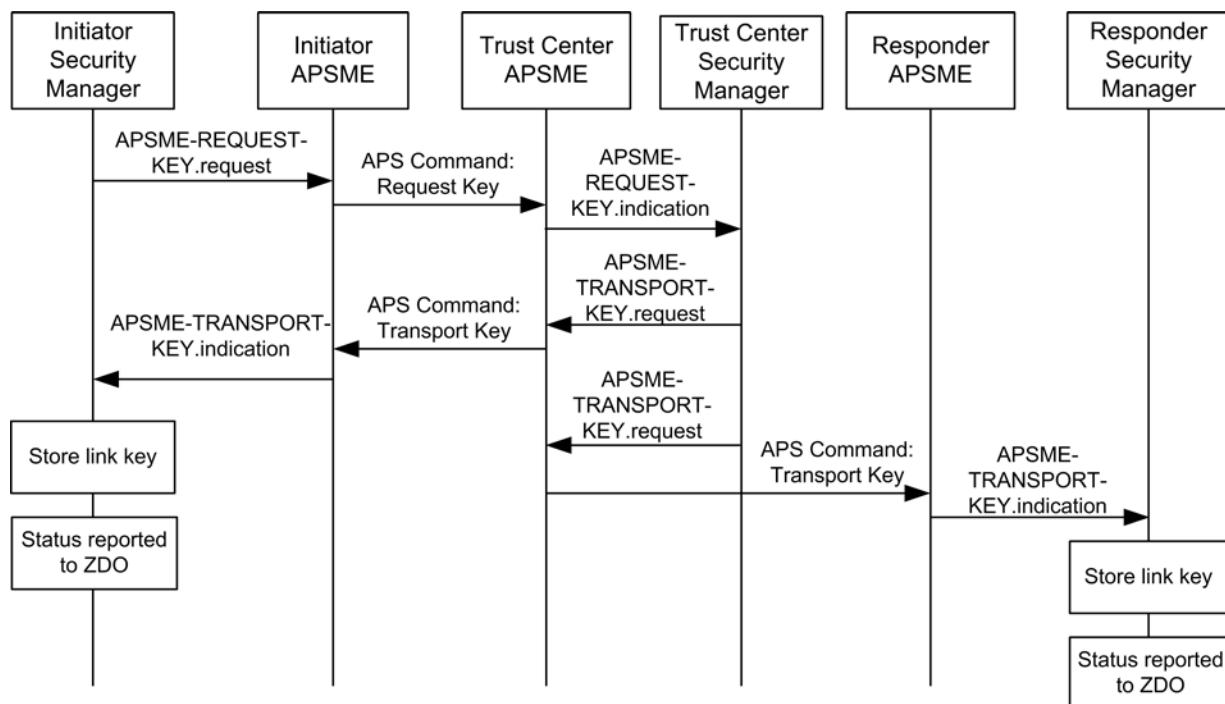
4.6.3.5.3 Message Sequence Chart

11497 An example message sequence chart of the end-to-end application key establishment procedure is shown
11498 Figure 4.26. The procedure begins with the transmission of the request-key command from the initiator to
11499 the Trust Center.

11500 The Trust Center shall now send transport-key commands containing the application link to the initiator
11501 and responder devices. Upon completion (or time-out), the status of the protocol is reported to the ZDOs of
11502 the initiator and responder devices. If successful, the initiator and responder will now share a link key and
11503 secure communications will be possible.

11504

Figure 4.26 Example End-to-End Application Key Establishment Procedure



11505

4.6.3.6 Network Leave

A device, its router, and the Trust Center shall follow the procedures described in this section when the device is to leave the network.

11509

4.6.3.6.1 Trust Center Operation

If a Trust Center wants a device to leave and if the Trust Center is not the router for that device, the Trust Center shall issue the APSME-REMOVE-DEVICE.request primitive, with the ParentAddress parameter set to the router's address and the ChildAddress parameter set to the address of the device it wishes to leave the network.

The Trust Center will also be informed of devices that leave the network. Upon receipt of an APSME-UPDATE-DEVICE.indication primitive with the Status parameter set to 0x02 (that is, device left), the DeviceAddress parameter shall indicate the address of the device that left the network and the SrcAddress parameter shall indicate the address of parent of this device.

11518

4.6.3.6.2 Router Operation

Routers are responsible for receiving remove-device commands and for sending update-device commands.

Upon receipt of an APSME-REMOVE-DEVICE.indication primitive, if the SrcAddress parameter is equal to the *apsTrustCenterAddress* attribute of the AIB then the command shall be accepted. The router shall ignore APSME-REMOVE-DEVICE.indication primitives with the SrcAddress parameter not equal to the *apsTrustCenterAddress* attribute of the AIB.

If the DeviceAddress corresponds to the local device's address, then the device shall remove itself from the network according to section 4.6.3.6.3. If the DeviceAddress corresponds to address of a child device then a router shall issue an NLME-LEAVE.request primitive with the DeviceAddress parameter the same as the DeviceAddress parameter of the APSME-REMOVE-DEVICE.indication primitive and the rejoin parameter set to 0. Other fields are defined by the stack profile.

11530 If the DeviceAddress does not correspond to the local device address, nor does it correspond to a child de-
11531 vice of the router, the command shall be discarded.

11532 Upon receipt of an NLME-LEAVE.indication primitive with the DeviceAddress parameter set to one of its
11533 children and with the Rejoin Parameter = 0, a router that is not also the Trust Center shall issue an
11534 APSME-UPDATE-DEVICE.request primitive with:

- 11535 • The DstAddress parameter set to the address of the Trust Center.
11536 • The Status parameter set to 0x02 (that is, device left).
11537 • The DeviceAddress parameter set to the DeviceAddress parameter of the
11538 NLME-LEAVE.indication primitive.

11539 If the router is the Trust Center, it should simply operate as the Trust Center and shall not issue the
11540 APSME-UPDATE-DEVICE.request primitive (see section 4.6.3.6.1).

4.6.3.6.3 Leaving Device Operation

11542 Devices are responsible for receiving and sending leave messages. The following rules apply to all three
11543 types of leave messages: NWK Leave Command, ZDO Mgmt Leave, and APS Command: Remove De-
11544 vice.

11545 In a secured ZigBee network, leave messages shall be secured with the active network key and sent with
11546 security enabled at the level indicated by the *nwkSecurityLevel* attribute in the NIB.

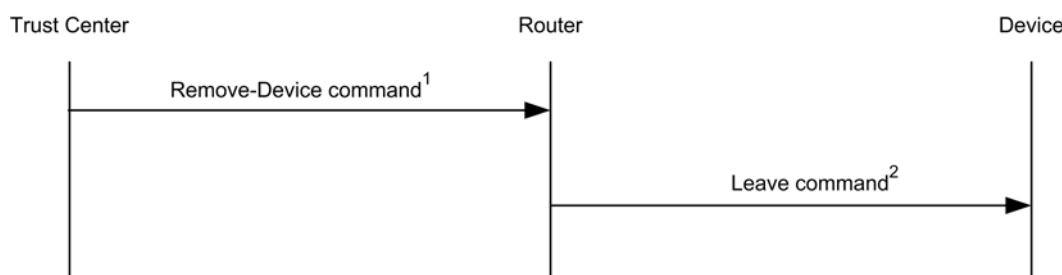
11547 In a secured ZigBee network, leave messages shall be received and processed only if secured with the ac-
11548 tive network key and received with security enabled at the level indicated by the *nwkSecurityLevel* attribute
11549 in the NIB.

11550 A device shall only send a NWK leave message (request or announcement) if it has the active network key.
11551 A device that wishes to leave the network and does not have the active network key shall quietly leave the
11552 network without sending a NWK leave announcement.

4.6.3.6.4 Message Sequence Charts

11554 Figure 4.27 shows an example message sequence chart in which a Trust Center asks a router to remove one
11555 of its children from the network. If a Trust Center wants a device to leave and if the Trust Center is not the
11556 router for that device, the Trust Center shall send the router a remove-device command with the address of
11557 the device it wishes to leave the network. In a secure network, the remove-device command shall be se-
11558 cured with a link key if present; otherwise shall be secured with the active network key. Upon receipt of the
11559 remove-device command, a router shall send a leave command to the device to leave the network.

11560 **Figure 4.27 Example Remove-Device Procedure**



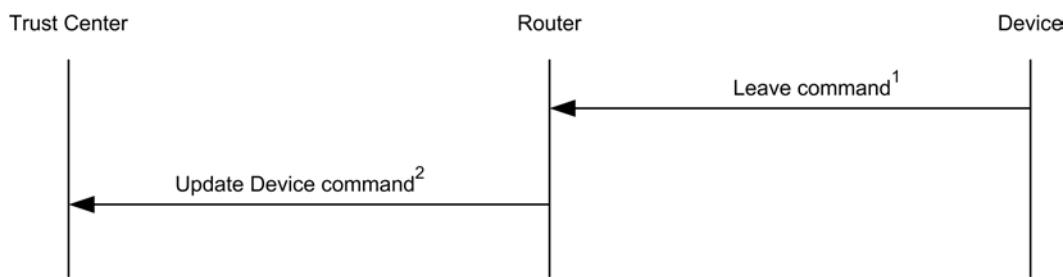
11561 **Notes:**

1. If a trust center wants a device to leave and if the trust center is not the router for that device, the trust center shall send the router a remove-device command with the address of the device it wishes to leave the network.
2. A router shall send a leave command to cause one of its children to leave the network.

11562 Figure 4.28 shows an example message sequence chart whereby a device notifies its router that it is leaving
11563 the network. In this example, the device sends a leave command (secured with the active network key) to
11564 its router. The router then sends an update-device command to the Trust Center. In a secured network, the
11565 update-device command must be secured with the link key, if present, or the active network key.

11566

Figure 4.28 Example Device-Leave Procedure



Notes:

1. A device leaving the network shall send a leave command to its router.
2. Upon receipt of a valid leave command, a router shall send an update-device command to the trust center to inform that a device has left the network.

11567

11568

11569

4.6.3.7 Command Tunneling

Devices shall follow the procedures described in this section to allow secure communication between the Trust Center and a remote device that does not have the current network key.

11572

4.6.3.7.1 Trust Center Operation

To embed a command in a tunnel command, the Trust Center shall first apply security protection as specified in section 4.4.1.1 and then, if security processing succeeds, the secured command frame shall be embedded in a Tunnel command frame as follows:

1. The APS header fields shall be set to the values of the APS header fields of the command to be embedded.
2. The destination address field shall be set to the 64-bit extended address of the destination device.
3. The tunneled auxiliary frame field shall be set to the auxiliary frame of the secured command, with following changes:
 - The extended nonce sub-field shall be set to 1;
 - The source address field shall be set to the 64-bit extended address of the Trust Center;
 - The tunneled command shall be set to the secured payload of the embedded command.

The tunneled command shall then be sent to the parent or other neighbor of the destination device.

4.6.3.7.2 Parent Operations

Upon receipt of an APS tunnel command, a router shall extract the embedded command as follows:

1. The APS header fields shall be set to the values of the APS header fields of the tunnel command.
2. The auxiliary frame field shall be set to the value of the tunneled auxiliary frame field of the tunnel command.
3. The APS payload field shall be set to the tunneled command field of the tunnel command.

The extracted command shall be sent to the destination indicated by the destination address field by issuing the NLDE-DATA.request primitive with security disabled.

4.6.3.7.3 Tunneled Data Destination Operation

The following applies to the end destination of the tunneled data payload after the parent has extracted and transmitted the payload from the APS tunnel command. Upon receipt of a message secured at the APS layer and with an extended nonce in the APS auxiliary frame, the message shall be processed as usual, except that the message shall not be looked up in, or added to, the APS duplicate rejection table.

11598

11599 4.7 Security Operations in Centralized Security Net- 11600 works

11601 The security services provided here offer a range of options within a ZigBee network. For interoperable
11602 and consistent field behavior, a more defined set of policies and processes is defined here. The basis for
11603 these operations is that the device forming a network can establish security policies believed appropriate for
11604 the network and that a joining device will acknowledge and use the policies in place in the network.
11605 Joining is therefore based on the forming device setting policies within the allowed settings in this section
11606 and the joining device having the appropriate flexibility to adapt to these settings.

11607 4.7.1 Trust Center Policies

11608 The Trust Center is a critical security component in a ZigBee network. The policies that the Trust Center
11609 puts in place control what devices get on the network and how they do so in a secure manner. Security is
11610 not an end unto itself but a means to establish a reasonable level of protection of the application and data
11611 that is being transmitted across the ZigBee network. Often an increase in security increases the overhead
11612 in management, requires additional time and functional states while security is negotiated, and can detract
11613 from a user experience by requiring them to go through additional steps that seem unnecessary. Therefore
11614 a balance must be struck between the hardening the network against attacks and the ability to use the net-
11615 work for the applications it was intended for.

11616 It is important to understand the security decisions that are being made in the network and the design of the
11617 Trust Center application is at the heart of those decisions. This section presents the options and settings
11618 for the Trust Center and requires a series of choices to be set on network start up.

11619 4.7.2 Trust Center Link Keys

11620 Support for link keys shall be required for all devices. Link keys offer an additional level of security for
11621 devices to be able to send messages with end-to-end security instead of just with the hop-by-hop security
11622 provided by network encryption.

11623 In addition, link keys are crucial for providing a simple authorization mechanism. The Trust Center can
11624 send devices a copy of the network key that is intended only for a specific device using that device's link
11625 key to encrypt the message.

11626

11627 4.7.3 Trust Center Policy Values

11628 The following is a list of configuration values that relate to the Trust Center's policy decisions that are part
11629 of the security related AIB in Table 4.29. They will be used to describe requirements for dictating the
11630 network security policies. The trust center can use these policies to create higher or lower sets of security
11631 and controls on the network. For example:

- 11632 • A system can be set up with centralized security such that any device can join the network. In
11633 such a permissive network, trust center link keys are still updated from the global value used ini-
11634 tially for joining.

- 11635 • A system can also be set up with trust center policies that only allow known devices. A user
 11636 must then install the IEEE address and a link key for the new device into the trust center prior to
 11637 the device joining. This could be done using install code based keys. This validates to the
 11638 joining device that it is on a network that knows its identity during the joining process. The trust
 11639 center in this network can also update the trust center link keys of joining devices so secure key
 11640 updates and rejoining can be conducted during the lifetime of the network.

11641 Table 4.32 describes the Trust Center policy values *trustCenterPolicies* of the AIB and their usage.

11642

11643

Table 4.33 Trust Center Policy Values

Attribute	Identifier	Type	Range	Description	Usage
<i>allowJoins</i>	0xad	bool- ean	TRUE or FALSE	This boolean indicates whether the Trust Center is currently allowing devices to join the network. A value of TRUE means that the Trust Center is allowing devices that have never been sent the network key or a trust center link key, to join the network.	This is set to FALSE in centralized security networks that do not want to allow new devices on the network.
<i>useWhite- List</i>	0xae	bool- ean	TRUE or FALSE	This boolean indicates whether the Trust Center allows any device with any IEEE to join or allows only known devices. A value of FALSE means that the Trust Center will allow any IEEE address to join the network. A value of TRUE means that the Trust Center will only allow IEEE addresses listed in <i>apsDeviceKey- PairSet</i> to join the network.	This is set to TRUE in centralized security networks that only allow devices known to them to join or rejoin. Trust centers that set this to TRUE shall provide a user interface or out of bands means to update the trust center with IEEE address of new devices to join the network.
<i>allow- Install- Codes</i>	0xaf	enum- eration	0x00 – 0x10	This enumeration indicates if the Trust Center requires install codes to be used with joining devices. 0x00 – do not support Install Codes 0x01 – Support but do not require use of Install Codes 0x02 – Require the use of Install Codes by joining devices	This is set to 0x02 if the trust center requires install codes in new devices. If this is set to 0x02 then useWhiteList would normally be set to TRUE. Trust Centers that support setting 0x01 or 0x02 shall provide a user interface or out of band means to input the Install Code.
<i>up- dateTrustC enter- LinkKeysR equired</i>	0xb3	Bool- ean	TRUE or FALSE	This boolean indicates whether or not devices are required to attempt to update their Trust Center Link Keys after joining. If set to TRUE, the device must attempt a procedure to update its link key after joining the network.	This is set to TRUE in centralized security networks.
<i>allow- Rejoins</i>	0xb6	Bool- ean	TRUE or FALSE	This value indicates if the trust center allows rejoins using well known or default keys. A setting	This is set to FALSE in centralized security networks.

				of FALSE means rejoins are only allowed with trust center link keys where the KeyAttributes of the <i>apsDeviceKeyPairSet</i> entry indicates VERIFIED_KEY.	
<i>allow-TrustCenterLinkKeyRequests</i>	0xb7	enum-eration	0x00 – 0x02	This value controls whether devices are allowed to request a Trust Center Link Key after they have joined the network. It may have the following values: 0x00 – never 0x01 – any device may request 0x02 – Only devices in the <i>apsDeviceKeyPairSet</i> with a KeyAttribute value of PROVISIONAL_KEY may request.	This is set to 0x00 in networks with higher level protocols for establishing link keys. This is set to either 0x01 or 0x02 in centralized security networks.
<i>network-KeyUpdatePeriod</i>	0xb9	Integer	0x00000000 – 0xFFFFFFFF	The period, in minutes, of how often the network key is updated by the Trust Center. A period of 0 means the Trust Center will not periodically update the network key (it may still update key at other times).	This is used in the Trust Center of centralized security networks to establish the network key update period. When this time is up the Trust Center updates the network key.
<i>network-KeyUpdateMethod</i>	0xba	enum-eration	0x00 – 0x01	This value describes the method the Trust Center uses to update the network key. 0x00 – Broadcast using only network encryption 0x01 – Unicast using network encryption and APS encryption with a device's link key.	This is used in centralized security networks to establish the policy for updating the network key.
<i>allowApplicationKeyRequests</i>	0xbb	enum-eration	0x00 – 0x02	This value determines how the Trust Center shall handle attempts to request an application link key with a partner node. 0x00 – never 0x01 – Any device may request an application link key with any device (except the Trust Center) 0x02 – Only those devices listed in <i>applicationKeyRequestList</i> may request and receive application link keys.	This is used in centralized security networks to establish the Trust Center policy on providing Application Link keys between devices on the network. It is normally set to 0x01 allowing any device to request a link key with another device to support those applications that want to encrypt application payload.
<i>applicationKeyRequestList</i>	0xbc	List of address pairs	Variable	This is a list of IEEE pairs of devices, which are allowed to establish application link keys between one another. The first IEEE address is the initiator, the second is the responder. If the responder address is set to 0xFFFFFFFFFFFFFF, then the initiator is allowed to request an application link key with any device. If the responder's address is	This list is normally not used in centralized security networks unless the Trust Center policy restricts those devices allowed to request link keys.

				not 0xFFFFFFFFFFFFFF, then it may also initiate an application link key request. This list is only valid if <i>allowApplicationkeyRequests</i> is set to 0x02.	
<i>allow-RemoteTcPolicyChange</i>	0xbd	Boolean	TRUE or FALSE	This policy indicates whether a node on the network that transmits a ZDO Mgmt_Permit_Join with a significance set to 1 is allowed to effect the local Trust Center's policies. ³	

11644

4.7.3.1 Allowing Devices to Join

If the Trust Center receives notification that a device is joining the network via the APSME-UPDATE-DEVICE.indication with the Status field set to Standard device unsecured join (0x01), the following procedure shall be performed:

1. If *allowJoins* is set to FALSE, the following shall be done.
 - a. The Trust Center shall proceed to the process of rejecting the join described in section 4.7.3.4. No further processing shall be done.
2. If *useWhiteList* is set to TRUE, the following shall be done.
 - a. Search the *apsDeviceKeyPairSet* for an address that matches the IEEE of the joining device. If none is found, it shall proceed to the process of rejecting the join described in section 4.7.3.4. No further processing shall be done.
3. The device has been authorized for admission to the network and the following shall be performed.
 - a. Examine *apsDeviceKeyPairSet* for an address that matches the IEEE of the joining device, if none is found the following shall be performed.
 - i. Add a new entry setting the *DeviceAddress* of the *apsDeviceKeyPairSet* to the *DeviceAddress* of the APSME-UPDATE-DEVICE.indication and the *LinkKey* of the *apsDeviceKeyPairSet* to the value 5A 69 67 42 65 65 41 6C 6C 69 61 6E 63 65 30 39 (ZigBeeAlliance09).
 - ii. Generate an APSME-TRANSPORT-KEY.request primitive with the following parameters.
 - i. Set the *DestAddress* to the *DeviceAddress* of the APSME-UPDATE-DEVICE.indication.
 - ii. Set the *StandardKeyType* to Standard Network Key (0x01).
 - iii. Set the *TransportKeyData* to the *Key* field of the active network key entry in the *nwkSecurityMaterialSet* NIB attribute.

4.7.3.2 Remote Device Changing Trust Center Policy

In some networks it may be permissible for a joined device to request that the Trust Center allow an unjoined device to be commissioned on the network. This can be accomplished through the ZDO Mgmt_Permit_Join_Req sent to the Trust Center with the TC_Significance field set to 1. Upon receipt of this request, the following procedure shall be executed.

³ CCB 1550

- 11675 1. If allowRemoteTcPolicyChange is set to 0, then the operation shall be denied and the status of
11676 0xa3 (ILLEGAL_REQUEST) passed back to the ZDO. No further processing shall be done.
11677 2. If useWhiteList is set to TRUE, then the operation is invalid and the status of 0xaa
11678 NOT_SUPPORTED) shall be passed back to the ZDO. No further processing shall be done.
11679 3. The operation is allowed by the Trust Center and a status of 0x00 (SUCCESS) shall be passed
11680 back to the ZDO.

11681
11682 When the new device requests to join the network the trust center will still process the joining device as
11683 described in section 4.7.3.1.⁴

11685 **4.7.3.3 Determining the Link Key for Encryption or Decryption 11686 by the Trust Center**

11687 If the Trust Center has determined that a message shall be sent with APS encryption or has been received
11688 and must be decrypted, it must determine what link key to use for the operation. The Trust Center shall
11689 examine the IEEE address of the destination (if encrypting) or source (if decrypting) and search the
11690 *apsDeviceKeyPairSet* for a matching address entry. If a match is found, it will use the associated link key
11691 to APS encrypt or decrypt the message.

11692 If no matching entry is found then no link key is defined and processing of the message shall be stopped.
11693 The message will not be sent or received because there is no link key that can be used.

11694 See sections 4.4.1.1 and 4.4.1.2 for incoming and outgoing frame processing.

11695 **4.7.3.4 Rejecting the Join or Rejoin**

11696 A join or rejoin is processed via an APSME-UPDATE-DEVICE.indication. Following the decision to re-
11697 ject a join or rejoin the following shall be done by the Trust Center.

- 11698 1. If the Status of APSME-UPDATE-DEVICE.indication was Standard Device Unsecured Join (0x01) or
11699 Standard Device Trust Center Rejoin (0x03), the following shall be done.
11700 a. The joining or rejoining device does not have the current network key and will be left to
11701 timeout.
11702 b. No further processing shall be done.
11703 2. If the Status of the APSME-UPDATE-DEVICE.indication was Standard Device Secured Rejoin
11704 (0x00), the following shall be done.
11705 a. Follow the procedure in section 4.7.3.5.

11707 **4.7.3.5 Removing Devices**

11708 The Trust Center has the ability to remove devices in the network via the APS Remove Device command.
11709 This message can be used to force well-behaved devices to leave the network. This is useful if the Trust
11710 Center determines after they have joined that they are not on the correct network or that the device is un-
11711 able to communicate in a required application specific way.

11712 It is important to note that this is not a secure means of removing a device. Once a malicious device has
11713 the current network key the only way to force it off the network is to distribute a new network key in a
11714 manner that prevents the malicious device from obtaining the new key. See section 4.7.3.10.

⁴ CCB 1550

11716 **4.7.3.6 Processing Trust Center Link Key Requests**

11717 The Trust Center link key is a crucial element in joining the network when a preconfigured key is in place,
11718 or when a device attempts to rejoin after a missed network key update. It is also the means by which
11719 application keys are established with other devices on the network. The process in ZigBee for transporting a
11720 new link key to the device requires the previous link key as an authentication mechanism. In addition it
11721 uses APS commands which do not have support for APS retries. As a result it is possible for devices to
11722 get out of sync with regard to the Trust Center link key currently in use. To avoid this risk the Trust Cen-
11723 ter may decide to prohibit requests for new trust center link keys when one is already in place.

11724 The following describes the process when the Trust Center is notified of an APS Request key via the
11725 APSME-REQUEST-KEY.indication with the RequestKeyType set to 0x04 (Trust Center Link Key):

- 11727 1. If the APS Command Request Key message is not APS encrypted, the device shall drop the message
11728 and no further processing shall take place.
- 11729 2. The device shall verify the key used to encrypt the APS command. If the SrcAddress of the
11730 APSME-REQUEST-KEY.indication primitive does not equal the value of the DeviceAddress of the
11731 corresponding apsDeviceKeyPairSet entry used to decrypt the message, the message shall be dropped
11732 and no further processing shall take place.
- 11733 3. If the RequestKeyType is set to 0x04, Trust Center Link Key, the following shall be performed:
 - 11734 a. If *allowTrustCenterLinkKeyRequests* is 0, then no more processing is done. The request is
11735 silently rejected.
 - 11736 b. If *allowTrustCenterLinkKeyRequests* is 1, then the following is performed:
 - 11737 i. Follow the procedure in section 4.7.3.6.1.
 - 11738 ii. If *allowTrustCenterLinkKeyRequests* is 2, do the following.
 - 11739 i. Find an entry in the apsDeviceKeyPairSet of the AIB where the DeviceAddress of
11740 the entry matches the PartnerAddress of the APSME-REQUEST-KEY.indication
11741 primitive, and the KeyAttributes has a value of PROVISIONAL_KEY (0x00). If
11742 no entry can be found matching those criteria, then the request shall be silently
11743 dropped and no more processing shall be done.
 - 11744 ii. Otherwise, follow the procedure in section 4.7.3.6.1.

11746 **4.7.3.6.1 Procedure for Generating and Sending a new Trust Center 11747 Link Key**

11748 This procedure takes an IEEE address DeviceAddress.

- 11749 1. Create a new 128-bit key, KeyValue. This may be done using a random number generator, or pro-
11750 grammatically using an algorithm.
- 11751 2. Create a new entry in the apsDeviceKeyPairSet.
 - 11752 a. Set the DeviceAddress of the entry to the DeviceAddress passed to this procedure.
 - 11753 b. Set the LinkKey value of the entry to the KeyValue previously generated in this procedure.
 - 11754 c. Set the KeyAttributes of the entry to UNVERIFIED_KEY (0x01).
 - 11755 d. Set the ApsLinkKeyType of the entry to Unique Link Key (0x00).
- 11756 3. If there is no space in the apsDeviceKeyPairSet attribute then processing shall fail and no further steps
11757 are executed in this procedure.

- 11758 4. Issue an APSME-TRANSPORT-KEY.request primitive with the DestAddress set to the DeviceAd-
11759 dress, the StandardKeyType set to 0x04 (Trust Center Link Key), and the TransportKeyData set to the
11760 KeyValue.

11761

11762

11763

11764 **4.7.3.7 Alternate methods of Updating the Trust Center Link**
11765 **Key**

11766 The process of using APS request key or unsolicited transport key messages for updating the Trust Center
11767 link key has several problems. The main problem is that of synchronization. Neither side knows whether
11768 or not the other side is now using the new key, and future attempts to update the key require knowledge
11769 of the current key that is being used.

11770 A better mechanism is a mutual authentication protocol that has the following properties:

1. The protocol must use one or more shared secrets that are not transmitted over the air during the
11772 protocol negotiation.
2. The protocol must allow both sides to inject random data in the key generation to prevent one de-
11774 vice from controlling the result of the key generation.
3. The protocol must not require knowledge of a previously generated Trust Center link key in order
11776 to generate a new one.

11777 The Certificate Based Key-Exchange has all of these properties.

11778 **4.7.3.8 Processing Application Link Key Requests**

11779 Devices may use the Trust Center to establish application link keys with one another. Those devices can
11780 leverage the secure communications channel they have established with the Trust Center in order to estab-
11781 lish secure communications with other devices. The Trust Center policy dictates whether or not it will
11782 answer application link key requests. Trust Center shall only allow application link key requests it re-
11783 ceives that are encrypted with the device's Trust Center link key. Any application link key request that is
11784 not APS encrypted shall be dropped. In addition, if the Trust Center does not have a link key in
11785 *apsDeviceKeyPairSet* for the responder device listed in the APS Request Key Command, it shall drop the
11786 request. The purpose of the using the Trust Center to establish an application link key is leverage the trust
11787 each device has with the Trust Center (through their Trust Center Link Key).

11788 The Trust Center shall ignore any requests made to establish application link keys with itself. ZigBee
11789 provides no protocol mechanism to differentiate whether a Trust Center link key or an application link key
11790 was used to encrypt a message. Therefore a device cannot determine what key to use when decrypting the
11791 message.

11792 It is worth noting that devices are not required to use the Trust Center to establish application link keys, and
11793 that some application profiles allow devices to establish link keys without the trust center. The applica-
11794 tion profile in use by the device may require that the Trust Center be utilized to do this.

11795 Application link key requests are initiated by the requesting device may occur at any time. Therefore the
11796 Trust Center shall not change its handling of those requests based on whether it is currently operating in
11797 commissioning or operational mode.

11798 Upon receipt of an APSME-REQUEST-KEY.indication with the RequestKeyType set to 0x02 (Application
11799 Link Key) the following shall be performed:

1. If the PartnerAddress of the primitive is equal to the *apsTrustCenterAddress* of the AIB, the re-
11801 quest shall be dropped and no further processing shall be done.

- 11802 2. If the Trust Center policy of allowApplicationLinkKeyRequests is 0x00, then the request shall be
11803 dropped and no further processing shall be done.
- 11804 3. If the Trust Center policy of allowApplicationLinkKeyRequests is 0x01, then the Trust Center
11805 shall do the following.
 - 11806 a. Run the procedure in section 4.7.3.8.1 using SrcAddress from the primitive as the InitiatorAddress in the procedure, and PartnerAddress from the primitive as the ResponderAddress in the procedure.
 - 11809 b. No further processing shall be done after that.
- 11810 4. If the Trust Center policy of allowApplicationLinkKeyRequests is 0x02, then the following shall
11811 be performed.
 - 11812 a. Find an entry in the allowApplicationKeyRequestList where the SrcAddress of the primitive matches the Initiator Address of the entry, and the PartnerAddress of the primitive matches the Responder Address of the entry.
 - 11815 b. If no entry is found, then the request shall be dropped and no further processing done.
 - 11816 c. If an entry is found, follow the procedure in section 4.7.3.8.1.

11818 **4.7.3.8.1 Procedure for Generating and Sending Application Link Keys**

11819 This procedure takes two IEEE addresses, InitiatorAddress and ResponderAddress.

- 11820 1. The Trust Center shall generate a random 128-bit key KeyValue for the application link key.
- 11821 2. It shall issue an APSME-TRANSPORT-KEY.request with the StandardKeyType set to 0x03, Application Link Key, the TransportKeyData set to KeyValue, and the DestAddress set to InitiatorAddress.
- 11823 3. It shall issue a sceond APSME-TRANSPORT-KEY.request with the StandardKeyType set to 0x03, Application Link Key, the TransportKeyData set to KeyValue, and the DestAddress set to ResponderAddress.

11828 **4.7.3.9 Key Lifetime**

11829 How long a network key or trust-center link key remains in use is up to the trust center. The longer a key
11830 is in use the more chance there is of it becoming compromised. On the other hand, updating a key too often
11831 adds management overhead and increases the risk that problems during key transmission will disrupt
11832 the network. A balance must be struck between the needs of security and the temporary disruption a new
11833 key can cause.

11834 **4.7.3.9.1 Link Key Lifetime**

- 11835 • It is advisable that the trust center have a policy for link keys to be changed periodically. This is
11836 can be difficult for sleepy end devices, which must check with the trust center periodically to receive
11837 any newly-available key.
- 11838 • It is recommended that old, unused link keys be deleted from the Trust Center to prevent them
11839 from being used. This requires that devices periodically communicate with the Trust Center using
11840 APS security to allow it to keep track of usage of the keys.
- 11841 • Often a link key is used to initially join the network and thus it is uncertain how long the key may
11842 have been in use before joining the network. Preconfigured link keys may be extremely long
11843 lived and thus increases the need to update the link key as soon as the device joins the network.

- 11844 • Link keys that are established using higher level protocols are not updated based on trust center
11845 policies but on the higher level application policies.

4.7.3.9.2 Network Key Lifetime

11847 The trust center shall periodically distribute and then switch to a new network key. There are two main
11848 reasons for doing this:

- 11849 1. An update and switch resets the outgoing NWK frame counter of all devices on the network.
11850 This lengthens the life of the network, since once the frame counter of a device gets to all
11851 0xFFFFFFFF it cannot send network encrypted traffic.
- 11852 2. It reduces the risk of a network key being compromised through attacks

11853 If a trust center detects that the frame counter for any device in its neighbor table is greater than
11854 0x80000000 it should update the network key.

11855 Trust centers should update the network key at least once per year. It is not recommended to update the
11856 network key more than once every 30 days except when required by the application or profile.

11857 Trust centers that do not have a real time clock or other means of tracking time are recommended to per-
11858 form a network key update when their outgoing frame counter reaches 0x40000000.

4.7.3.10 Updating the Network Key

11860 Updating the Network key is one of the core responsibilities of the Trust Center. It helps to insure that a
11861 key does not remain in use for too long and thus is not too susceptible to compromise.

4.7.3.10.1 Period of Updates

11863 The network key shall be updated periodically. How often an update is sent out is based on the *nwkKey-*
11864 *TrustCenterUpdatePeriod*.

4.7.3.10.2 Sleepy Devices

11866 Sleepy devices may miss many network events, such as a channel change, PAN ID change, or a parent that
11867 has left. Sleepy devices may not be awake at the point when the Trust Center is updating the network
11868 key, regardless of whether the key is broadcast or unicast. If the sleeping device happens to poll within
11869 nwkTransactionPersistenceTime for a unicast key update, or nwkBroadcastDeliveryTime for a broadcast
11870 key update, the update message shall be delivered. Otherwise the delivery of the key update to the sleepy
11871 device will timeout and the sleepy device will not receive the update.

11872 The sleepy device should consider the network key update another one of those events and will need to
11873 handle that case when waking up. A child that sends a message to its parent and receives a MAC ack but
11874 no response at the application layer may have missed a key update and therefore should try to perform a
11875 rejoin. If the parent has switched to the newer key, the sleeping device must perform a trust center rejoin.

4.7.3.10.3 Broadcast Network Key Updates

11877 Broadcast key updates are the simplest mechanism for distributing new network keys. The new network
11878 key is broadcast using the existing network key to encrypt it. There is no way to exclude a device that has
11879 the current network key from this kind of key update.

4.7.3.10.4 Unicast Network Key Updates

11881 A more secure way of sending out network key updates is by using unicast messages encrypted with each
11882 device's link key. This requires that all authorized devices on the network have a link key so that the
11883 Trust Center can individually update them in a secure manner. A Trust Center that wishes to securely re-
11884 move a previously authorized device should use this mechanism as it can be used to exclude a device from
11885 the network.

1186 If this unicast method is used by the trust center, it is required that the Trust Center maintain a list of all the
1187 routers on the network and send key updates to only those devices. Sleepy devices are unlikely to be
1188 awake when the Trust Center decides to change the network key. Sending to only routers also reduces the
1189 amount of network traffic that the Trust Center has to generate in order to update the network.

11890 **4.7.3.10.5 Key Switch**

11891 Regardless of the mechanism used to perform a key update (broadcast or unicast), it is required that the
11892 APS key switch command be broadcast. Devices will implicitly switch the network key when they hear
11893 another device using the newer key. This mechanism insures that even if the device did not receive the
11894 formal key switch, it will start using the new key

11895 A device can determine if the new network key is actively being used by examining the key sequence
11896 number in the NWK auxiliary header of packets it receives. If it receives a message that passes decryption
11897 using the new key sequence number then it shall switch to using the newer network key and stop
11898 sending message encrypted using the old network key.

11899 **4.7.3.10.6 Old Network Keys**

11900 A network key update and switch does not preclude the use of the previous network key. A device is al-
11901 lowed to accept messages encrypted using the last network key, as this insures a smooth transition to the
11902 new key. A device shall never send messages using the old key.

11903 To completely deprecate a key's use, the Trust Center will have to perform an update and switch twice. If
11904 using a broadcast key update, the Trust Center should make sure that both the key update and the key
11905 switch broadcasts have completely expired before sending a second set to update and switch.

11907 **4.8 Security Operations in Distributed Security Net- 11908 works**

11909
11910 In distributed security networks, there is not a single trust center in control of the network. Each router
11911 can act as a parent and transport keys to joining devices. In addition, if a device already has a network
11912 key from an out of band installation method or commissioning, the device is accepted into the network
11913 without any trust center authorization.

11914 **4.8.1 Trust Center Address**

11915 In distributed security networks the trust center address is 0xFFFFFFFFFFFFFF. This address is used
11916 in transport key commands as the source address to indicate the network is in a distributed security model.

11917 **4.8.2 Network Key Updates**

11918 Network key updates are not done in distributed security networks.

11919 **4.8.3 Link Keys**

11920 Link keys are only used to APS encrypt transport key commands during joining in distributed security
11921 networks. The key type stored internally shall be 0x01 (Global Link Key).

4.8.4 Application Link Keys

11922 Devices may require use of application link keys for APS data. In a distributed security network the partner devices must use a higher level protocol to establish the application link key without the trust center involvement or permissions.

4.8.5 Requesting Keys

11926 There is no facility to process or answer APSME-REQUEST-KEY primitives. All APS Command Request Key frames shall be dropped and no further processing shall be done.

4.9 Device Operations

11930 Devices joining the network shall also have policies that dictate what security they expect from the network.
11931 The following are the settings that can be used to adjust their security policy.

4.9.1 Joining Device Policy Values

11933 A joining device may have a set of policy values enumerated in Table 4.33. However, it normally sets these
11934 policy values upon joining based on if the network is a centralized or distributed security model. All
11935 devices shall support joining either network and adapting their security policies accordingly unless their
11936 application profile mandates joining only one type of network.

11937
11938 **Table 4.34 Joining Device Policy Values**

Name	Type	Range	Description	Usage
<i>requestNewTrustCenterLinkKey</i>	Boolean	TRUE or FALSE	This boolean indicates whether the device will request a new Trust Center Link key after joining. A value of TRUE means the device shall send an APS request key command to the Trust Center with Request-KeyType 0x04. If the request is not answered <i>requestLinkKeyTimeout</i> seconds then the device will leave the network. A value of FALSE means the device will not request a new link key.	This is set to TRUE in centralized security networks to ensure devices have a trust center link key for rejoining or key updates. Note this value is set to FALSE in a distributed security network.
<i>requestLinkKeyTimeout</i>	Integer	0 – 10	This integer indicates the maximum time in seconds that a device will wait for a response to a request for a Trust Center link key.	This is ignored in a distributed security network.
<i>acceptNewUnsolicitedApplicationLinkKey</i>	Boolean	TRUE or FALSE	This boolean indicates whether the device will accept a new unsolicited application link key sent to it by the Trust Center.	

11939

4.9.2 Trust Center Address

11940
11941
11942
11943

A device will not know the address of the Trust Center prior to joining. The *apsTrustCenterAddress* in the AIB shall be initially set to 0x0000000000000000. Upon joining a device shall receive an APS Transport key and the source address shall indicate the address of the trust center. The *apsTrustCenterAddress* in the AIB will be set to the address in the received packet.

11944
11945

A value of 0xFFFFFFFFFFFFFF for the *apsTrustCenterAddress* in the AIB indicates a distributed security network and the device settings should be adjusted accordingly.

11946
11947

See section 4.4.1.5 for a description of when and how the trust center address of APS commands are validated.

11948

4.9.3 Trust Center Link Keys

11949
11950
11951
11952
11953

All devices in a centralized security network shall obtain an updated Trust Center link key when they first join the network and the Trust Center supports this behavior. An updated trust center link key protects the device from compromise if the original joining key is discovered. The application may utilize a key establishment algorithm if one is available. If such an algorithm is not available, the Request Key services of the APSME must be used.

11954
11955
11956
11957

Prior to revision 21 of this specification, there was not an interoperable mechanism to update the link key so. Therefore a Trust Center operating on a prior revision is not assumed to have support for this behavior. Determining the Trust Center revision can be done using the Server Mask and the ZDO Node Descriptor Request. Initiation of this process is done by the higher application.

11958
11959

Once the device has obtained an updated Trust Center link key it shall ignore any APS commands from the Trust Center that are not encrypted with that key.

11960

4.9.4 Receiving new Link Keys

11961
11962
11963
11964

It is possible a device's security policy may restrict application link keys sent to it by the trust center for use with another partner device. This could be because the device wishes to control which other devices it shares link keys with, or because it uses some other mechanism to establish application link keys with devices besides the trust center.

11965
11966
11967

There are instances where higher level application policies determine what data is shared with application link keys. For example, networks where updated Trust Center link keys must be established through the Certificate Based Key Exchange protocol.

11968
11969

If the device receives a transport key command containing an application link key, but it has not sent a request for one, and *acceptNewUnsolicitedApplicationLinkKey* is set to FALSE, it shall ignore the message.

11970

4.9.5 Requesting a Link Key

11971
11972
11973
11974
11975

A device shall attempt to update its trust center link key as part of its initial joining operations in a centralized security network. Trust Centers prior to the revision 21 version of this specification did not support updating trust center link keys via the APSME request key method. Determination of whether the trust center supports this behavior is left up to the higher level application. The application may use either the APSME Request Key facilities or an alternative key establishment protocol.

11976
11977
11978

If the device is requesting a trust center link key using the APSME, it shall start a timer after sending the initial request. Once the timer has reached *requestLinkKeyTimeout*, the device shall no longer accept a transport key message containing a new Trust Center link key unless the device initiates a new request.

11979 If the device is requesting an application link key and acceptNewUnsolicitedApplicationLinkKey is set to
11980 FALSE, it shall start a timer after sending the initial request. Once the timer has reached re-
11981 questLinkKeyTimeout the device shall no longer accept a transport key message containing a new applica-
11982 tion link key unless it initiates a new request.

11983 A device that did not request a new application link key and has acceptNewUnsolicitedApplicationLinkKey
11984 set to FALSE shall silently drop the APS Transport Key Command for an application link key. It shall
11985 not process the command.

11986

11987

11988

11989

11990

11991

11992

11993

11994

11995

11996

11997

11998

11999

12000

12001

12002

12003

This page intentionally left blank.

ANNEX A

CCM* MODE OF OPERATION

CCM* is a generic combined encryption and authentication block cipher mode. CCM* is only defined for use with block ciphers having a 128-bit block size, such as AES-128 [B8]. The CCM* principles can easily be extended to other block sizes, but doing so will require further definitions.

The CCM* mode coincides with the original CCM mode specification [B21] for messages that require authentication and, possibly, encryption, but does also offer support for messages that require only encryption. As with the CCM mode, the CCM* mode requires only one key. The security proof for the CCM mode([B22] and [B23]) carries over to the CCM* mode described here. The design of the CCM* mode takes into account the results of [B24], thus allowing it to be securely used in implementation environments in which the use of variable-length authentication tags, rather than fixed-length authentication tags only, is beneficial.

Prerequisites: The following are the prerequisites for the operation of the generic CCM* mode:

1. A block-cipher encryption function E shall have been chosen, with a 128-bit block size. The length in bits of the keys used by the chosen encryption function is denoted by $keylen$.
2. A fixed representation of octets as binary strings shall have been chosen (for example, most-significant-bit first order or least-significant-bit-first order).
3. The length L of the message length field, in octets, shall have been chosen. Valid values for L are the integers 2, 3,..., 8 (the value $L=1$ is reserved).
4. The length M of the authentication field, in octets, shall have been chosen. Valid values for M are the integers 0, 4, 6, 8, 10, 12, 14, and 16. (The value $M=0$ corresponds to disabling authenticity, since then the authentication field contains an empty string.)

A.1 Notation and Representation

Throughout this specification, the representation of integers as octet strings shall be fixed. All integers shall be represented as octet strings in most-significant-octet first order. This representation conforms to the conventions in Section 4.3 of ANSI X9.63-2001 [B7].

A.2 CCM* Mode Encryption and Authentication Transformation

The CCM* mode forward transformation involves the execution, in order, of an input transformation (A.2.1), an authentication transformation (A.2.2), and encryption transformation (A.2.3).

Input: The CCM* mode forward transformation takes as inputs:

1. A bit string Key of length $keylen$ bits to be used as the key. Each entity shall have evidence that access to this key is restricted to the entity itself and its intended key-sharing group member(s).
2. A nonce N of 15- L octets. Within the scope of any encryption key Key , the nonce value shall be unique.
3. An octet string m of length $l(m)$ octets, where $0 \leq l(m) \leq 28L$.
4. An octet string a of length $l(a)$ octets, where $0 \leq l(a) < 2^{64}$.

The nonce N shall encode the potential values for M such that one can uniquely determine from N the value of M actually used. The exact format of the nonce N is outside the scope of this specification and shall be determined and fixed by the actual implementation environment of the CCM* mode.

12044 Note: The exact format of the nonce N is left to the application, to allow simplified hardware and software imple-
12045 ments in particular settings. Actual implementations of the CCM* mode may restrict the values of M that are
12046 allowed throughout the life-cycle of the encryption key Key to a strict subset of those allowed in the generic CCM*
12047 mode. If so, the format of the nonce N shall be such that one can uniquely determine from N the actually used value
12048 of M in that particular subset. In particular, if M is fixed and the value $M=0$ is not allowed, then there are no re-
12049 strictions on N , in which case the CCM* mode reduces to the CCM mode.

12050 **A.2.1 Input Transformation**

12051 This step involves the transformation of the input strings a and m to the strings $AuthData$ and $PlainText-$
12052 $Data$, to be used by the authentication transformation and the encryption transformation, respectively.

12053 This step involves the following steps, in order:

- 12054 1. Form the octet string representation $L(a)$ of the length $l(a)$ of the octet string a , as follows:
 - 12055 a. If $l(a)=0$, then $L(a)$ is the empty string.
 - 12056 b. If $0 < l(a) < 2^{16}-2^8$, then $L(a)$ is the 2-octets encoding of $l(a)$.
 - 12057 c. If $2^{16}-2^8 \leq l(a) < 2^{32}$, then $L(a)$ is the right-concatenation of the octet 0xff, the octet 0xfe, and the
12058 4-octets encoding of $l(a)$.
 - 12059 d. If $2^{32} \leq l(a) < 2^{64}$, then $L(a)$ is the right-concatenation of the octet 0xff, the octet 0xff, and the
12060 8-octets encoding of $l(a)$.
- 12061 2. Right-concatenate the octet string $L(a)$ with the octet string a itself. Note that the resulting string con-
12062 tains and a encoded in a reversible manner.
- 12063 3. Form the padded message $AddAuthData$ by right-concatenating the resulting string with the smallest
12064 non-negative number of all-zero octets such that the octet string $AddAuthData$ has length divisible by
12065 16.
- 12066 4. Form the padded message $PlaintextData$ by right-concatenating the octet string m with the smallest
12067 non-negative number of all-zero octets such that the octet string $PlaintextData$ has length divisible by
12068 16.
- 12069 5. Form the message $AuthData$ consisting of the octet strings $AddAuthData$ and $PlaintextData$:

$$12070 AuthData = AddAuthData // PlaintextData$$

12071 **A.2.2 Authentication Transformation**

12072 The data $AuthData$ that was established above shall be tagged using the tagging transformation as follows:

- 12073 1. Form the 1-octet $Flags$ field consisting of the 1-bit $Reserved$ field, the 1-bit $Adata$ field, and the 3-bit
12074 representations of the integers M and L , as follows:

$$12075 Flags = Reserved // Adata // M // L$$

12076 Here, the 1-bit $Reserved$ field is reserved for future expansions and shall be set to ‘0’. The 1-bit $Adata$
12077 field is set to ‘0’ if $l(a)=0$, and set to ‘1’ if $l(a)>0$. The L field is the 3-bit representation of the integer
12078 $L-1$, in most-significant-bit-first order. The M field is the 3-bit representation of the integer $(M-2)/2$ if
12079 $M>0$ and of the integer 0 if $M=0$, in most-significant-bit-first order.

- 12080 2. Form the 16-octet B_0 field consisting of the 1-octet $Flags$ field defined above, the 15- L octet nonce
12081 field N , and the L -octet representation of the length field $l(m)$, as follows:

$$12082 B_0 = Flags // Nonce N // l(m)$$

- 12083 3. Parse the message $AuthData$ as $B_1 // B_2 // \dots // B_t$, where each message block B_i is a 16-octet string.

12084 The CBC-MAC value X_{t+1} is defined by:

$$12085 X_0 := 0_{128}; X_{i+1} := E(Key, X_i \oplus B_i) \text{ for } i=0, \dots, t.$$

12086 Here, $E(K, x)$ is the cipher-text that results from encryption of the plaintext x using the established
12087 block-cipher encryption function E with key Key ; the string 0^{128} is the 16-octet all-zero bit string.

12088 The authentication tag T is the result of omitting all but the leftmost M octets of the CBC-MAC value
12089 X_{n+1} thus computed.

12090 A.2.3 Encryption Transformation

12091 The data *PlaintextData* that was established in section A.2.1 (step 4) and the authentication tag T that was
12092 established in section A.2.2 (step 3) shall be encrypted using the encryption transformation as follows:

- 12093 1. Form the 1-octet *Flags* field consisting of two 1-bit *Reserved* fields, and the 3-bit representations of
12094 the integers O and L , as follows:

$$Flags = Reserved // Reserved // O // L$$

12096 Here, the two 1-bit *Reserved* fields are reserved for future expansions and shall be set to ‘0’. The L
12097 field is the 3-bit representation of the integer $L-1$, in most-significant-bit-first order. The ‘0’ field is
12098 the 3-bit representation of the integer 0, in most-significant-bit-first order.

12099 Define the 16-octet A_i field consisting of the 1-octet *Flags* field defined above, the $15-L$ octet nonce
12100 field N , and the L -octet representation of the integer i , as follows:

$$A_i = Flags // Nonce N // Counter i, for i=0, 1, 2, \dots$$

12102 Note that this definition ensures that all the A_i fields are distinct from the B_0 fields that are actually
12103 used, as those have a *Flags* field with a non-zero encoding of M in the positions where all A_i fields
12104 have an all-zero encoding of the integer 0 (see section A.2.2, step 1).

12105 Parse the message *PlaintextData* as $M_1 // \dots // M_t$, where each message block M_i is a 16-octet string.

12106 The ciphertext blocks C_1, \dots, C_t are defined by:

$$C_i := E(Key, A_i) \oplus M_i \text{ for } i=1, 2, \dots, t$$

12108 The string *Ciphertext* is the result of omitting all but the leftmost $l(m)$ octets of the string $C_1 // \dots // C_t$

12109 Define the 16-octet encryption block S_0 by:

$$S_0 := E(Key, A_0)$$

- 12111 2. The encrypted authentication tag U is the result of XOR-ing the string consisting of the leftmost M octets
12112 of S_0 and the authentication tag T .

12113 **Output:** If any of the above operations has failed, then output ‘invalid’. Otherwise, output the
12114 right-concatenation of the encrypted message *Ciphertext* and the encrypted authentication tag U .

12115 A.3 CCM* Mode Decryption and Authentication 12116 Checking Transformation

12117 **Input:** The CCM* inverse transformation takes as inputs:

- 12118 1. A bit string *Key* of length *keylen* bits to be used as the key. Each entity shall have evidence that access
12119 to this key is restricted to the entity itself and its intended key-sharing group member(s).
- 12120 2. A nonce N of 15- L octets. Within the scope of any encryption key *Key*, the nonce value shall be
12121 unique.
- 12122 3. An octet string c of length $l(c)$ octets, where $0 \leq l(c)-M < 2^{8L}$.
- 12123 4. An octet string a of length $l(a)$ octets, where $0 \leq l(a) < 2^{64}$.

12124 A.3.1 Decryption Transformation

12125 The decryption transformation involves the following steps, in order:

- 12126 1. Parse the message c as $C // U$, where the rightmost string U is an M -octet string. If this operation fails,
12127 output ‘invalid’ and stop. U is the purported encrypted authentication tag. Note that the leftmost string
12128 C has length $l(c)-M$ octets.

- 12129 2. Form the padded message *CiphertextData* by right-concatenating the string *C* with the smallest
12130 non-negative number of all-zero octets such that the octet string *CiphertextData* has length divisible by
12131 16.
12132 3. Use the encryption transformation in section A.2.3, with the data *CipherTextData* and the tag *U* as in-
12133 puts.
12134 4. Parse the output string resulting from applying this transformation as $m \parallel T$, where the rightmost string
12135 *T* is an *M*-octet string. *T* is the purported authentication tag. Note that the leftmost string *m* has length
12136 $l(c) \cdot M$ octets.

12137 **A.3.2 Authentication Checking Transformation**

12138 The authentication checking transformation involves the following steps:

- 12139 1. Form the message *AuthData* using the input transformation in section A.2.1, with the string *a* and the
12140 octet string *m* that was established in section A.3.1 (step 4) as inputs.
12141 2. Use the authentication transformation in section A.2.2, with the message *AuthData* as input.
12142 3. Compare the output tag *MACTag* resulting from this transformation with the tag *T* that was established
12143 in section A.3.1 (step 4). If *MACTag*=*T*, output ‘valid’; otherwise, output ‘invalid’ and stop.

12144 **Output:** If any of the above verifications has failed, then output ‘invalid’ and reject the octet string *m*.
12145 Otherwise, accept the octet string *m* and accept one of the key sharing group member(s) as the source of *m*.

12146 **A.4 Restrictions**

12147 All implementations shall limit the total amount of data that is encrypted with a single key. The CCM* en-
12148 cryption transformation shall invoke not more than 2^{61} block-cipher encryption function operations in total,
12149 both for the CBC-MAC and for the CTR encryption operations.

12150 At CCM* decryption, one shall verify the (truncated) CBC-MAC before releasing any information, such as,
12151 *Plaintext*. If the CBC-MAC verification fails, only the fact that the CBC-MAC verification failed shall be
12152 exposed; all other information shall be destroyed.

12153

ANNEX B SECURITY BUILDING BLOCKS

12154

12155
12156

This annex specifies the cryptographic primitives and mechanisms that are used to implement the security protocols in this standard.

12157
12158

B.1 Symmetric-Key Cryptographic Building Blocks

12159
12160

The following symmetric-key cryptographic primitives and data elements are defined for use with all security-processing operations specified in this standard.

12161

B.1.1 Block-Cipher

12162
12163
12164

The block-cipher used in this specification shall be the Advanced Encryption Standard AES-128, as specified in FIPS Pub 197. This block-cipher has a key size *keylen* that is equal to the block size, in bits, *i.e.*, *keylen*=128.

12165

B.1.2 Mode of Operation

12166
12167

The block-cipher mode of operation used in this specification shall be the CCM* mode of operation, as specified in section A.2.3, with the following instantiations:

12168
12169
12170
12171

1. Each entity shall use the block-cipher *E* as specified in section B.1.1.
2. All octets shall be represented as specified in the “Conventions and Abbreviations.”
3. The parameter *L* shall have the integer value 2.
4. The parameter *M* shall have one of the following integer values: 0, 4, 8, or 16.

12172

B.1.3 Cryptographic Hash Function

12173
12174

The cryptographic hash function used in this specification shall be the blockcipher based cryptographic hash function specified in section B.6, with the following instantiations:

12175
12176

1. Each entity shall use the block-cipher *E* as specified in section B.1.1.
2. All integers and octets shall be represented as specified in section 1.2.1.

12177
12178

The Matyas-Meyer-Oseas hash function (specified in section B.6) has a message digest size *hashlen* that is equal to the block size, in bits, of the established block-cipher.

12179 **B.1.4 Keyed Hash Function for Message Authen-**
12180 **tication**

12181 The keyed hash message authentication code (HMAC) used in this specification shall be HMAC,
12182 as specified in the FIPS Pub 198 [B9], with the following instantiations:

- 12183 1. Each entity shall use the cryptographic hash H function as specified in section B.1.3..
- 12184 2. The block size B shall have the integer value 16 (this block size specifies the length of the data
12185 integrity key, in bytes, that is used by the keyed hash function, *i.e.*, it uses a 128-bit data
12186 integrity key).
- 12187 3. The output size $HMAClen$ of the HMAC function shall have the same integer value as the
12188 message digest parameter $hashlen$ as specified in section B.1.3.

12189 **B.1.5 Specialized Keyed Hash Function for Mes-**
12190 **sage Authentication**

12191 The specialized keyed hash message authentication code used in this specification shall be as
12192 specified in section B.1.4.

12193 **B.1.6 Challenge Domain Parameters**

12194 The challenge domain parameters used in the specification shall be as specified in section B.3.1,
12195 with the following instantiation: ($minchallengelen, maxchallengelen$)=(128,128).

12196 All challenges shall be validated using the challenge validation primitive as specified in section
12197 B.4.

12198 **B.2 Key Agreement Schemes**

12199 **B.2.1 Symmetric-Key Key Agreement Scheme**

12200 The symmetric-key key agreement protocols in this standard shall use the full symmetric-key with
12201 key confirmation scheme, with the following instantiations:

- 12202 1. Each entity shall use the HMAC-scheme as specified in section B.1.4.
- 12203 2. Each entity shall use the specialized HMAC-scheme as specified in section B.1.5.
- 12204 3. Each entity shall use the cryptographic hash function as specified in section B.1.3.
- 12205 4. The parameter $keydatalen$ shall have the same integer value as the key size parameter $keylen$
12206 as specified in section B.1.1.
- 12207 5. The parameter $SharedData$ shall be the empty string; parameter $shareddatalen$ shall have the
12208 integer value 0.
- 12209 6. The optional parameters $Text_1$ and $Text_2$ as specified in section B.7.1 and section B.7.2 shall
12210 both be the empty string.
- 12211 7. Each entity shall use the challenge domain parameters as specified in section B.1.6.
- 12212 8. All octets shall be represented as specified in section 1.2.1.

12213 **B.3 Challenge Domain Parameter Generation and** 12214 **Validation**

12215 This section specifies the primitives that shall be used to generate and validate challenge domain
12216 parameters.

12217 Challenge domain parameters impose constraints on the length(s) of bit challenges a scheme ex-
12218 pects. As such, this establishes a bound on the entropy of challenges and, thereby, on the security
12219 of the cryptographic schemes in which these challenges are used. In most schemes, the challenge
12220 domain parameters will be such that only challenges of a fixed length will be accepted (for exam-
12221 ple, 128-bit challenges). However, one may define the challenge domain parameters such that
12222 challenges of varying length might be accepted. Doing so is useful in contexts in which entities
12223 that wish to engage in cryptographic schemes might have a bad random number generator
12224 onboard. Allowing both entities that engage in a scheme to contribute sufficiently long inputs ena-
12225 bles each of them to contribute sufficient entropy to the scheme.

12226 In this standard, challenge domain parameters will be shared by a number of entities using a
12227 scheme determined by the standard. The challenge domain parameters may be public; the security
12228 of the system does not rely on these parameters being secret.

12229 **B.3.1 Challenge Domain Parameter Generation**

12230 Challenge domain parameters shall be generated using the following routine.

12231 **Input:** This routine does not take any input.

12232 **Actions:** The following actions are taken:

- 12233 1. Choose two nonnegative integers *minchallengelen* and *maxchallengelen*, such that *minchal-*
12234 *lengelen* ≤ *maxchallengelen*.

12235 **Output:** Challenge domain parameters *D*=(*minchallengelen*, *maxchallengelen*).

12236 **B.3.2 Challenge Domain Parameter Verification**

12237 Challenge domain parameters shall be verified using the following routine.

12238 **Input:** Purported set of challenge domain parameters *D*=(*minchallengelen*, *maxchallengelen*).

12239 **Actions:** The following checks are made:

- 12240 1. Check that *minchallengelen* and *maxchallengelen* are non-negative integers.
12241 2. Check that *minchallengelen* ≤ *maxchallengelen*.

12242 **Output:** If any of the above verifications has failed, then output ‘invalid’ and reject the challenge
12243 domain parameters. Otherwise, output ‘valid’ and accept the challenge domain parameters.

12244 **B.4 Challenge Validation Primitive**

12245 It is used to check whether a challenge to be used by a scheme in the standard has sufficient length
12246 (for example, messages that are too short are discarded, due to insufficient entropy).

12247 **Input:** The input of the validation transformation is a valid set of challenge domain parameters
12248 *D*=(*minchallengelen*, *maxchallengelen*), together with the bit string *Challenge*.

12249 **Actions:** The following actions are taken:

- 12250 1. Compute the bit-length *challengelen* of the bit string *Challenge*.

12251 2. Verify that $challengelen \in [minchallengelen, maxchallengelen]$. (That is, verify that the chal-
12252 lenge has an appropriate length.)

12253 **Output:** If the above verification fails, then output ‘invalid’ and reject the challenge. Otherwise,
12254 output ‘valid’ and accept the challenge.

12255 B.5 Secret Key Generation (SKG) Primitive

12256 This section specifies the SKG primitive that shall be used by the symmetric-key key agreement
12257 schemes specified in this standard.

12258 This primitive derives a shared secret value from a challenge owned by an entity U_1 and a chal-
12259 lenge owned by an entity U_2 when all the challenges share the same challenge domain parameters.
12260 If the two entities both correctly execute this primitive with corresponding challenges as inputs,
12261 the same shared secret value will be produced.

12262 The shared secret value shall be calculated as follows:

12263 **Prerequisites:** The following are the prerequisites for the use of the SKG primitive:

- 12264 1. Each entity shall be bound to a unique identifier (e.g., distinguished names).
12265 All identifiers shall be bit strings of the same length $entlen$ bits. Entity U_1 ’s identifier will be
12266 denoted by the bit string U_1 . Entity U_2 ’s identifier will be denoted by the bit string U_2 .
12267 2. A specialized MAC scheme shall be chosen, with tagging transformation as specified in Sec-
12268 tion 5.7.1 of ANSI X9.63-2001 [B7]. The length in bits of the keys used by the specialized
12269 MAC scheme is denoted by $mackeylen$.

12270 **Input:** The SKG primitive takes as input:

- 12271 • A bit string $MACKey$ of length $mackeylen$ bits to be used as the key of the established spe-
12272 cialized MAC scheme.
12273 • A bit string QEU_1 owned by U_1 .
12274 • A bit string QEU_2 owned by U_2 .

12275 **Actions:** The following actions are taken:

- 12276 1. Form the bit string consisting of U_1 ’s identifier, U_2 ’s identifier, the bit string QEU_1 corre-
12277 sponding to U_1 ’s challenge, and the bit string QEU_2 corresponding to QEU_2 ’s challenge:
12278 $MacData = U_1 // U_2 // QEU_1 // QEU_2$
12279 2. Calculate the tag $MacTag$ for $MacData$ under the key $MACKey$ using the tagging transfor-
12280 mation of the established specialized MAC scheme:
12281 $MacTag = MACMacKey(MacData)$
12282 3. If the tagging transformation outputs ‘invalid’, output ‘invalid’ and stop.
12283 4. Set $Z=MacTag$.

12284 **Output:** The bit string Z as the shared secret value.

12285 B.6 Block-Cipher-Based Cryptographic Hash 12286 Function

12287 This section specifies the Matyas-Meyer-Oseas hash function, a cryptographic hash function based
12288 on block-ciphers. We define this hash function for blockciphers with a key size equal to the block
12289 size, such as AES-128, and with a particular choice for the fixed initialization vector IV (we take
12290 $IV=0$). For a more general definition of the Matyas-Meyer-Oseas hash function, refer to Section
12291 9.4.1 of [B19].

12292 **Prerequisites:** The following are the prerequisites for the operation of Matyas- Meyer-Oseas hash
12293 function:

- 12294 1. A block-cipher encryption function E shall have been chosen, with a key size that is equal to
12295 the block size. The Matyas-Meyer-Oseas hash function has a message digest size that is equal
12296 to the block size of the established encryption function. It operates on bit strings of length less
12297 than 2^{2n} , where n is the block size, in octets, of the established block-cipher.
- 12298 2. A fixed representation of integers as binary strings or octet strings shall have been chosen.

12299 **Input:** The input to the Matyas-Meyer-Oseas hash function is as follows:

- 12300 1. A bit string M of length l bits, where $0 \leq l < 2^{2n}$

12301 **Actions:** The hash value shall be derived as follows:

- 12302 1. If the message M has length less than 2^n bits, pad this message according to the following
12303 procedure:

- 12304 a. Right-concatenate to the message M the binary consisting of the bit '1' followed by k '0'
12305 bits, where k is the smallest non-negative solution to the equation:

$$l+1+k \equiv 7n \pmod{8n} \quad (1)$$

- 12306 b. Form the padded message M' by right-concatenating to the resulting string the n -bit
12307 string that is equal to the binary representation of the integer l .

- 12308 2. Otherwise pad this message according to the following method:

- 12309 a. Right concatenate to the message M the binary consisting of the bit '1' followed by k '0'
12310 bits, where k is the smallest non-negative solution to the equation:

$$l + 1 + k \equiv^1 5n \pmod{8n} \quad (2)$$

- 12311 b. Form the padded message M' by right-concatenating to the resulting string the $2n$ -bit
12312 string that is equal to the binary representation of the integer l and right-concatenating to
12313 the resulting string the n -bit all-zero bit string.

- 12314 3. Parse the padded message M' as $M_1 // M_2 // \dots // M_t$, where each message block M_i is an n -octet
12315 string.

- 12316 4. The output $Hash_t$ is defined by

$$Hash_0 = 0^{8n}; Hash_j = E(Hash_{j-1}, M_j) \oplus M_j \text{ for } j=1, \dots, t \quad (3)$$

12317 Here, $E(K, x)$ is the ciphertext that results from encryption of the plaintext x , using the estab-
12318 lished block-cipher encryption function E with key K ; the string 0^{8n} is the n -octet all-zero bit
12319 string.

12320 **Output:** The bit string $Hash_t$ as the hash value.

12321 Note that the cryptographic hash function operates on bit strength of length less than 2^{2n} bits,
12322 where n is the block size (or key size) of the established block cipher, in bytes. For example, the
12323 Matyas-Meyer-Oseas hash function with AES- 128 operates on bit strings of length less than 232
12324 bits. It is assumed that all hash function calls are on bit strings of length less than 2^{2n} bits. Any
12325 scheme attempting to call the hash function on a bit string exceeding 2^{2n} bits shall output 'invalid'
12326 and stop.

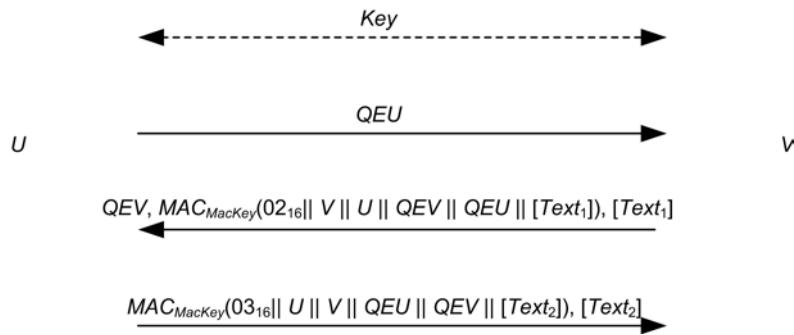
¹ CCB 1434

B.7 Symmetric-Key Authenticated Key Agreement Scheme

This section specifies the full symmetric-key key agreement with key confirmation scheme. A MAC scheme is used to provide key confirmation. Note that all key exchanges and random challenges shall be assumed within data strings in network transmission order.

Figure B.1 illustrates the messaging involved in the use of the full symmetric-key key agreement with key confirmation scheme.

Figure B.1 Symmetric-Key Authenticated Key Agreement Scheme



The scheme is ‘asymmetric’, so two transformations are specified. U uses the transformation specified in section B.7.1 to agree on keying data with V if U is the protocol’s initiator, and V uses the transformation specified in section B.7.2 to agree on keying data with U if V is the protocol’s responder. The essential difference between the role of the initiator and the role of the responder is that the initiator sends the first pass of the exchange.

If U executes the initiator transformation, and V executes the responder transformation with the shared secret keying material as input, then U and V will compute the same keying data.

Prerequisites: The following are the prerequisites for the use of the scheme:

1. Each entity has an authentic copy of the system’s challenge domain parameters $D=(minchallengelen, maxchallengelen)$.
2. Each entity shall have access to a bit string Key of length $keylen$ bits to be used as the key. Each party shall have evidence that access to this key is restricted to the entity itself and the other entity involved in the symmetric-key authenticated key agreement scheme.
3. Each entity shall be bound to a unique identifier (for example, distinguished names). All identifiers shall be bit strings of the same length $entlen$ bits. Entity U ’s identifier will be denoted by the bit string U . Entity V ’s identifier will be denoted by the bit string V .
4. Each entity shall have decided which MAC scheme to use as specified in Section 5.7 of ANSI X9.63-2001 [B7]. The length in bits of the keys used by the chosen MAC scheme is denoted by $mackeylen$.
5. A cryptographic hash function shall have been chosen for use with the key derivation function.
6. A specialized MAC scheme shall have been chosen for use with the secret key generation primitive with tagging transformation as specified in Section 5.7.1 of ANSI X9.63-2001 [B7]. The length in bits of the keys used by the specialized MAC scheme is denoted by $keylen$.
7. A fixed representation of octets as binary strings shall have been chosen. (for example, most-significant-bit-first order or least-significant-bit-first order).

12365

B.7.1 Initiator Transformation

12366
12367
12368

U shall execute the following transformation to agree on keying data with *V* if *U* is the protocol's initiator. *U* shall obtain an authentic copy of *V*'s identifier and an authentic copy of the static secret key *Key* shared with *V*.

12369
12370
12371
12372
12373

Input: The input to the initiator transformation is:

1. An integer *keydatalen* that is the length in bits of the keying data to be generated.
2. (Optional) A bit string *SharedData* of length *shareddatalen* bits that consists of some data shared by *U* and *V*.
3. (Optional) A bit string *Text*² that consists of some additional data to be provided from *U* to *V*.

12374
12375
12376
12377

Ingredients: The initiator transformation employs the challenge generation primitive specified in Section 5.3 of ANSI X9.63-2001 [B7], the challenge validation primitive in section B.3.2, the SKG primitive in section B.5, the key derivation function in Section 5.6.3 of ANSI X9.63-2001 [B7], and one of the MAC schemes in Section 5.7 of ANSI X9.63-2001 [B7].

12378

Actions: Keying data shall be derived as follows:

12379
12380
12381
12382
12383
12384
12385
12386
12387

1. Use the challenge generation primitive in Section 5.3 of ANSI X9.63-2001 [B7] to generate a challenge *QEU* for the challenge domain parameters *D*. Send *QEU* to *V*.
2. Then receive from *V* a challenge *QEV'*, purportedly owned by *V*. If this value is not received, output 'invalid' and stop.
3. Verify that *QEV'* is a valid challenge for the challenge domain parameters *D* as specified in section B.3.2. If the validation primitive rejects the challenge, output 'invalid' and stop.
4. Use the SKG primitive given in section B.5 to derive a shared secret bit string *Z* from the challenges *Q₁=QEU* owned by *U* and *Q₂=QEV'* owned by *V*, using as key the shared key *Key*. If the SKG primitive outputs 'invalid', output 'invalid' and stop.
5. Use the key derivation function in Section 5.6.3 of ANSI X9.63-2001 [B7] with the established hash function to derive keying data *KKeyData* of length *mackeylen+keydatalen* bits from the shared secret value *Z* and the shared data [*SharedData*].
6. Parse the leftmost *mackeylen* bits of *KKeyData* as a MAC key *MacKey* and the remaining bits as keying data *KeyData*.

12391
12392

7. Form the bit string consisting of the octet *02₁₆*, *V*'s identifier, *U*'s identifier, the bit string *QEV'*, the bit string *QEU*, and if present *Text₁*:

$$MacData_1 = 02_{16} \parallel V \parallel U \parallel QEV' \parallel QEU \parallel [Text_1]$$

12396
12397
12398
12399

8. Verify that *MacTag₁* is the tag for *MacData₁* under the key *MacKey* using the tag checking transformation of the appropriate MAC scheme specified in Section 5.7.2 of ANSI X9.63-2001 [B7]. If the tag checking transformation outputs 'invalid', output 'invalid' and stop.

12400
12401
12402

9. Form the bit string consisting of the octet *03₁₆*, *U*'s identifier, *V*'s identifier, the bit string *QEU* corresponding to *U*'s challenge, the bit string *QEV'* corresponding to *V*'s challenge, and optionally a bit string *Text₂*:

$$MacData_2 = 03_{16} \parallel U \parallel V \parallel QEU \parallel QEV' \parallel [Text_2]$$

12403
12404
12405

10. Calculate the tag *MacTag₂* on *MacData₂* under the key *MacKey* using the tagging transformation of the appropriate MAC scheme specified in Section 5.7.1 of ANSI X9.63-2001 [B7]:

$$MacTag_2 = MAC_{MacKey}(MacData_2)$$

12406
12407
12408

11. If the tagging transformation outputs 'invalid', output 'invalid' and stop. Send *MacTag₂* and, if present, *Text₂* to *V*.

12409
12410

12. Receive from *V* an optional bit string *Text₁*, and a purported tag *MacTag₁*. If these values are not received, output 'invalid' and stop.

(4)

12411 **Output:** If any of the above verifications has failed, then output ‘invalid’ and reject the bit strings
12412 *KeyData* and *Text_I*. Otherwise, output ‘valid’, accept the bit string *KeyData* as the keying data of
12413 length *keydatalen* bits shared with *V* and accept *V* as the source of the bit string *Text_I* (if present).

12414 B.7.2 Responder Transformation

12415 *V* shall execute the following transformation to agree on keying data with *U* if *V* is the protocol’s
12416 responder. *V* shall obtain an authentic copy of *U*’s identifier and an authentic copy of the static se-
12417 cret key *Key* shared with *U*.

12418 **Input:** The input to the responder transformation is:

- 12419 1. A challenge *QEU'* purportedly owned by *U*.
- 12420 2. An integer *keydatalen* that is the length in bits of the keying data to be generated.
- 12421 3. (Optional) A bit string *SharedData* of length *shareddatalen* bits that consists of some data
12422 shared by *U* and *V*.
- 12423 4. (Optional) A bit string *Text_I* that consists of some additional data to be provided from *V* to *U*.

12424 **Ingredients:** The responder transformation employs the challenge generation primitive specified
12425 in Section 5.3 of ANSI X9.63-2001 [B7], the challenge validation primitive specified in section
12426 B.3.2, the SKG primitive given in section B.5, the key derivation function in Section 5.6.3 of AN-
12427 SI X9.63-2001 [B7], and one of the MAC schemes in Section 5.7 of ANSI X9.63-2001 [B7].

12428 **Actions:** Keying data shall be derived as follows:

- 12429 1. Verify that *QEU'* is a valid challenge for the challenge domain parameters *D* as specified in
12430 section B.3.2. If the validation primitive rejects the challenge, output ‘invalid’ and stop.
- 12431 2. Use the challenge generation primitive in Section 5.3 of ANSI X9.63-2001 [B7] to generate a
12432 challenge *QEV* for the challenge domain parameters *D*. Send to *U* the challenge *QEV*.
- 12433 3. Then receive from *U* an optional bit string *Text₂* and a purported tag *MacTag₂*. If this data is
12434 not received, output ‘invalid’ and stop.
- 12435 4. Form the bit string consisting of the octet 03_{16} , *U*’s identifier, *V*’s identifier, the bit string
12436 *QEU'* corresponding to *U*’s purported challenge, the bit string *QEV* corresponding to *V*’s
12437 challenge, and the bit string *Text₂* (if present):
$$MacData_2 = 03_{16} \parallel U \parallel V \parallel QEU' \parallel QEV \parallel [Text_2]$$
- 12439 5. Verify that *MacTag₂* is the valid tag on *MacData₂* under the key *MacKey* using the tag
12440 checking transformation of the appropriate MAC scheme specified in Section 5.7 ANSI
12441 X9.63-2001 [B7]. If the tag checking transformation outputs ‘invalid’, output ‘invalid’ and
12442 stop.
- 12443 6. Use the SKG primitive in section B.5 to derive a shared secret bit string *Z* from the challenges
12444 *Q₁=QEU'* owned by *U* and *Q₂=QEV* owned by *V*, using as key the shared key *Key*. If the
12445 SKG primitive outputs ‘invalid’, output ‘invalid’ and stop.
- 12446 7. Use the key derivation function in Section 5.6.3 of ANSI X9.63-2001 [B7] with the estab-
12447 lished hash function to derive keying data *KKeyData* of length *mackeylen+keydatalen* bits
12448 from the shared secret value *Z* and the shared data [*SharedData*].
- 12449 8. Parse the leftmost *mackeylen* bits of *KKeyData* as a MAC key *MacKey* and the remaining bits
12450 as keying data *KeyData*.
- 12451 9. Form the bit string consisting of the octet 02_{16} , *V*’s identifier, *U*’s identifier, the bit string
12452 *QEV*, the bit string *QEU'*, and, optionally, a bit string *Text_I*:
$$MacData_1 = 02_{16} \parallel V \parallel U \parallel QEV \parallel QEU' \parallel [Text_I]$$
- 12454 10. Calculate the tag *MacTag₁* for *MacData₁* under the key *MacKey* using the tagging transfor-
12455 mation of the appropriate MAC scheme specified in Section 5.7 of ANSI X9.63-2001 [B7]:
$$MacTag_1 = MAC_{MacKey}(MacData_1)$$

12457 If the tagging transformation outputs ‘invalid’, output ‘invalid’ and stop. Send to U , if present the
12458 bit string $Text_1$, and $MacTag_1$.

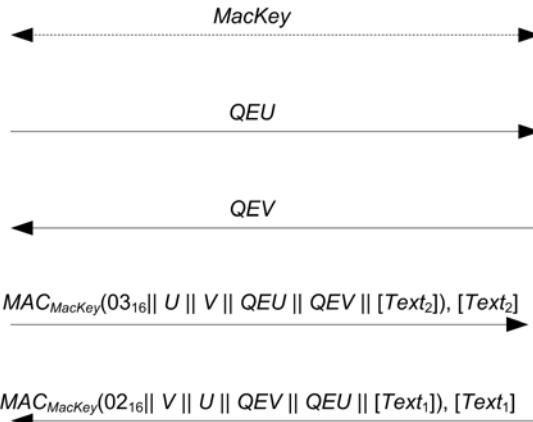
12459 **Output:** If any of the above verifications has failed, then output ‘invalid’ and reject the bit strings
12460 $KeyData$ and $Text_2$. Otherwise, output ‘valid’, accept the bit string $KeyData$ as the keying data of
12461 length $keydatalen$ bits shared with U and accept U as the source of the bit string $Text_2$ (if present).

B.8 Mutual Symmetric-Key Entity Authentication

12463 This section specifies the mutual symmetric-key entity authentication scheme. A MAC scheme is
12464 used to provide key confirmation.

12465 Figure B.2 illustrates the messaging involved in the use of mutual symmetric-key entity authen-
12466 tication scheme.

12467 **Figure B.2 Mutual Symmetric-Key Entity Authentication Scheme**



12468
12469 The scheme is ‘asymmetric’, so two transformations are specified. U uses the transformation speci-
12470 fied in section B.8.1 to establish authenticity of, and optionally obtain authenticated data from,
12471 V by means of sharing a key and communicating cooperatively with V . V uses the transformation
12472 specified in section B.8.2 to establish authenticity of, and optionally obtain authenticated data
12473 from, U by means of sharing a key and communicating cooperatively with U .

12474 The essential difference between the role of the initiator and the role of the responder is that the
12475 initiator sends the first pass of the exchange.

12476 **Prerequisites:** The following are the prerequisites for the use of the scheme:

- 12477 1. Each entity has an authentic copy of the system’s challenge domain parameters
12478 $D=(minchallengelen, maxchallengelen)$. These parameters shall have been generated using
12479 the parameter generation primitive in section B.3.1. Furthermore, the parameters shall have
12480 been validated using the parameter validation primitive in section B.3.2.
- 12481 2. Each entity shall have access to a bit string $MacKey$ of length $mackeylen$ bits to be used as the
12482 key. Each party shall have evidence that access to this key is restricted to the entity itself and
12483 the other entity involved in the mutual entity authentication scheme.
- 12484 3. Each entity shall be bound to a unique identifier (for example, distinguished names). All iden-
12485 tifiers shall be bit strings of the same length $entlen$ bits. Entity U ’s identifier will be denoted
12486 by the bit string U . Entity V ’s identifier will be denoted by the bit string V .
- 12487 4. Each entity shall have decided which MAC scheme to use as specified in Section 5.7 of ANSI
12488 X9.63-2001 [B7]. The length in bits of the keys used by the chosen MAC scheme is denoted
12489 by $mackeylen$.

- 12490 5. A fixed representation of octets as binary strings shall have been chosen (for example,
12491 most-significant-bit-first order or least-significant-bit-first order).

12492 **B.8.1 Initiator Transformation**

12493 *U* shall execute the following transformation to establish authenticity of, and optionally obtain authen-
12494 ticated data from, *V* by means of sharing a key and communicating cooperatively with *V*. *U* shall obtain an authentic copy of *V*'s identifier and an authentic copy of the secret key *MacKey* shared with *V*.

12497 **Input:** The input to the initiator transformation is:

- 12498 1. (Optional) A bit string *Text*₂ that consists of some additional data to be provided from *U* to *V*.

12499 **Ingredients:** The initiator transformation employs the challenge generation primitive specified in
12500 Section 5.3 of ANSI X9.63-2001 [B7], the challenge validation primitive specified in section B.4,
12501 and one of the MAC schemes in Section 5.7 of ANSI X9.63-2001 [B7].

12502 **Actions:** Entity authentication shall be established as follows:

- 12503 1. Use the challenge generation primitive given in Section 5.3 of ANSI X9.63- 2001 [B7] to
12504 generate a challenge *QEU* for the challenge domain parameters *D*. Send *QEU* to *V*.
- 12505 2. Then receive from *V* a challenge *QEV'* purportedly owned by *V*. If this value is not received,
12506 output 'invalid' and stop.
- 12507 3. Verify that *QEV'* is a valid challenge for the challenge domain parameters *D* as specified in
12508 section B.3.1. If the validation primitive rejects the challenge, output 'invalid' and stop.
- 12509 4. Form the bit string consisting of the octet 03₁₆, *U*'s identifier, *V*'s identifier, the bit string
12510 *QEU* corresponding to *U*'s challenge, the bit string *QEV'* corresponding to *V*'s purported
12511 challenge, and optionally a bit string *Text*₂:
$$MacData_2 = 03_{16} \parallel U \parallel V \parallel QEU \parallel QEV' \parallel [Text_2]$$
- 12512 5. Calculate the tag *MacTag*₂ on *MacData*₂ under the key *MacKey* using the tagging transfor-
12513 mation of the appropriate MAC scheme specified in Section 5.7.1 of ANSI X9.63-2001 [B7]:
$$MacTag_2 = MAC_{MacKey}(MacData_2)$$

If the tagging transformation outputs 'invalid', output 'invalid' and stop. Send *MacTag*₂ and, if present, bit string *Text*₂ to *V*.
- 12514 6. Receive from *V* an optional bit string *Text*₁, and a purported tag *MacTag*₁'. If these values are
12515 not received, output 'invalid' and stop.
- 12516 7. Form the bit string consisting of the octet 02₁₆, *V*'s identifier, *U*'s identifier, the bit string
12517 *QEV'* corresponding to *V*'s purported challenge, the bit string *QEU* corresponding to *U*'s
12518 challenge, and if present *Text*₁:
$$MacData_1 = 02_{16} \parallel V \parallel QEV' \parallel QEU \parallel [Text_1]$$
- 12519 8. Verify that *MacTag*₁' is the tag for *MacData*₁ under the key *MacKey* using the tag checking
12520 transformation of the appropriate MAC scheme specified in Section 5.7.2 of ANSI
12521 X9.63-2001 [B7]. If the tag checking transformation outputs 'invalid', output 'invalid' and
12522 stop.

12523 **Output:** If any of the above verifications has failed, then output 'invalid' and reject the authen-
12524 ticity of *V* and reject the entity authentication from *V*. Otherwise, output 'valid', accept the authen-
12525 ticity of *V* and accept the entity authentication from *V* of the authenticated bit string *Text*₁ (if pre-
12526 sent).

12532

B.8.2 Responder Transformation

12533
12534
12535
12536

V shall execute the following transformation to establish authenticity, of and optionally obtain authenticated data from, *U* by means of sharing a key and communicating cooperatively with *U*. *V* shall obtain an authentic copy of *U*'s identifier and an authentic copy of the secret key *MacKey* shared with *U*.

12537

Input: The input to the responder transformation is:

12538
12539
12540

1. A challenge *QEU'* purportedly owned by *U*.
2. (Optional) A bit string *Text₁* that consists of some additional data to be provided from *V* to *U*.

12541
12542
12543

Ingredients: The responder transformation employs the challenge generation primitive specified in Section 5.3 of ANSI X9.63-2001 [B7], the challenge validation primitive specified in section B.4, and one of the MAC schemes in Section 5.7 of ANSI X9.63-2001 [B7].

12544

Actions: Entity authentication shall be established as follows:

12545
12546
12547
12548
12549
12550

1. Verify that *QEU'* is a valid challenge for the challenge domain parameters *D* as specified in section B.3.1. If the validation primitive rejects the challenge, output 'invalid' and stop.
2. Use the challenge generation primitive in Section 5.3 of ANSI X9.63-2001 [B7] to generate a challenge *QEV* for the challenge domain parameters *D*. Send *QEV* to *U*.
3. Then receive from *U* an optional bit string *Text₂* and a purported tag *MacTag₂'*. If this data is not received, output 'invalid' and stop.

12551
12552
12553

$$MacData_2 = 03_{16} // U // V // QEU' // QEV // [Text_2]$$

12555
12556
12557

4. Form the bit string consisting of the octet 03_{16} , *U*'s identifier, *V*'s identifier, the bit string *QEU'* corresponding to *U*'s purported challenge, the bit string *QEV* corresponding to *V*'s challenge, and the bit string *Text₂* (if present):

$$MacData_2 = MAC_{MacKey}(MacData_2).$$

12558
12559
12560
12561
12562

If the tagging transformation outputs 'invalid', output 'invalid' and stop.

12563
12564
12565

5. Verify that *MacTag₂'* is the valid tag on *MacData₂* under the key *MacKey* using the tag checking transformation of the appropriate MAC scheme specified in Section 5.7.1 of ANSI X9.63-2001 [B7]. If the tag checking transformation outputs 'invalid', output 'invalid' and stop.

12566

6. Form the bit string consisting of the octet 02_{16} , *V*'s identifier, *U*'s identifier, the bit string *QEV* corresponding to *V*'s challenge, the bit string *QEU'* corresponding to *U*'s purported challenge and optionally a bit string *Text₁*:

$$MacData_1 = 02_{16} // V // U // QEV // QEU' // [Text_1].$$

12567
12568
12569

7. Calculate the tag *MacTag₁* for *MacData₁* under the key *MacKey* using the tagging transformation of the appropriate MAC scheme specified in Section 5.7.1 of ANSI X9.63-2001 [B7]:

$$MacTag_1 = MAC_{MacKey}(MacData_1).$$

12570
12571

If the tagging transformation outputs 'invalid', output 'invalid' and stop. Send *MacTag₁* and, if present, bit string *Text₁* to *U*.

12572
12573
12574
12575

Output: If any of the above verifications has failed, then output 'invalid' and reject the authenticity of *U* and reject the entity authentication from *U*. Otherwise, output 'valid', accept the authenticity of *U* and accept the entity authentication from *U* of the authenticated bit string *Text₂* (if present).

12576

12577

12578

12579

12580

12581

12582

12583

12584

12585

12586

12587

12588

12589

12590

12591

12592

12593

12594

12595

This page intentionally left blank.

ANNEX C

TEST VECTORS FOR CRYPTOGRAPHIC BUILDING BLOCKS

This annex provides sample test vectors for the ZigBee community, aimed at with the intent of assisting in building interoperable security implementations. The sample test vectors are provided as is, pending independent validation.

C.1 Data Conversions

For test vectors, see Appendix J1 of ANSI X9.63-2001 [B7].

C.2 AES Block Cipher

This annex provides sample test vectors for the block-cipher specified in section B.1.1.

For test vectors, see FIPS Pub 197 [B8].

C.3 CCM* Mode Encryption and Authentication Transformation

This annex provides sample test vectors for the mode of operation as specified in section B.1.2.

Prerequisites: The following prerequisites are established for the operation of the mode of operation:

1. The parameter M shall have the integer value 8.

Input: The inputs to the mode of operation are:

1. The key Key of size $keylen=128$ bits to be used:

$Key = C0\ C1\ C2\ C3\ C4\ C5\ C6\ C7\ C8\ C9\ CA\ CB\ CC\ CD\ CE\ CF$

2. The nonce N of $15-L=13$ octets to be used:

$Nonce = A0\ A1\ A2\ A3\ A4\ A5\ A6\ A7\ ||\ 03\ 02\ 01\ 00\ ||\ 06$

3. The octet string m of length $l(m)=23$ octets to be used:

$m = 08\ 09\ 0A\ 0B\ 0C\ 0D\ 0E\ 0F\ 10\ 11\ 12\ 13\ 14\ 15\ 16\ 17\ 18\ 19\ 1A\ 1B\ 1C\ 1D\ 1E$

4. The octet string a of length $l(a)=8$ octets to be used:

$a = 00\ 01\ 02\ 03\ 04\ 05\ 06\ 07$

12621 **C.3.1 Input Transformation**

12622 This step involves the transformation of the input strings a and m to the strings $AuthData$ and $PlainTextData$,
12623 to be used by the authentication transformation and the encryption transformation, respectively.

- 12624 1. Form the octet string representation $L(a)$ of the length $l(a)$ of the octet string a :

12625 $L(a) = 00\ 08$

- 12626 2. Right-concatenate the octet string $L(a)$ and the octet string a itself:

12627 $L(a) // a = 00\ 08 // 00\ 01\ 02\ 03\ 04\ 05\ 06\ 07$

- 12628 3. Form the padded message $AddAuthData$ by right-concatenating the resulting string with the smallest
12629 non-negative number of all-zero octets such that the octet string $AddAuthData$ has length divisible by 16:

12630 $AddAuthData = 00\ 08 // 00\ 01\ 02\ 03\ 04\ 05\ 06\ 07 // 00\ 00\ 00\ 00\ 00\ 00\ 00\ 00$

- 12631 4. Form the padded message $PlainTextData$ by right-concatenating the octet string m with the smallest
12632 non-negative number of all-zero octets such that the octet string $PlainTextData$ has length divisible by
12633 16:

12634 $PlainTextData = 08\ 09\ 0A\ 0B\ 0C\ 0D\ 0E\ 0F\ 10\ 11\ 12\ 13\ 14\ 15\ 16\ 17 //$
12635 $18\ 19\ 1A\ 1B\ 1C\ 1D\ 1E // 00\ 00\ 00\ 00\ 00\ 00\ 00\ 00$

- 12636 5. Form the message $AuthData$ consisting of the octet strings $AddAuthData$ and $PlainTextData$:

12637 $AuthData = 00\ 08\ 00\ 01\ 02\ 03\ 04\ 05\ 06\ 07\ 00\ 00\ 00\ 00\ 00\ 00 //$
12638 $08\ 09\ 0A\ 0B\ 0C\ 0D\ 0E\ 0F\ 10\ 11\ 12\ 13\ 14\ 15\ 16\ 17$
12639 $18\ 19\ 1A\ 1B\ 1C\ 1D\ 1E\ 00\ 00\ 00\ 00\ 00\ 00\ 00\ 00$

12640 **C.3.2 Authentication Transformation**

12641 The data $AuthData$ that was established above shall be tagged using the following tagging transformation:

- 12642 1. Form the 1-octet $Flags$ field as follows:

12643 $Flags = 59$

- 12644 2. Form the 16-octet B_0 field as follows:

12645 $B_0 = 59 // A0\ A1\ A2\ A3\ A4\ A5\ A6\ A7\ 03\ 02\ 01\ 00\ 06 // 00\ 17$

- 12646 3. Parse the message $AuthData$ as $B_1 // B_2 // B_3$, where each message block B_i is a 16-octet string.

12647

4. The CBC-MAC value X_4 is calculated as follows:

i	B_i	X_i
0	59 A0 A1 A2 A3 A4 A5 A6 A7 03 02 01 00 06 00 17	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
1	00 08 00 01 02 03 04 05 06 07 00 00 00 00 00 00	F7 74 D1 6E A7 2D C0 B3 E4 5E 36 CA 8F 24 3B 1A
2	08 09 0A 0B 0C 0D 0E 0F 10 11 12 13 14 15 16 17	90 2E 72 58 AE 5A 4B 5D 85 7A 25 19 F3 C7 3A B3
3	18 19 1A 1B 1C 1D 1E 00 00 00 00 00 00 00 00 00 00	5A B2 C8 6E 3E DA 23 D2 7C 49 7D DF 49 BB B4 09
4	∞	B9 D7 89 67 04 BC FA 20 B2 10 36 74 45 F9 83 D6

12648

The authentication tag T is the result of omitting all but the leftmost $M=8$ octets of the CBC-MAC value X_4 :

12649

$$T = B9 D7 89 67 04 BC FA 20$$

12650

C.3.3 Encryption Transformation

12651

The data *PlaintextData* shall be encrypted using the following encryption transformation:

12652

1. Form the 1-octet Flags field as follows:

12653

$$Flags = 01$$

12654

2. Define the 16-octet A_i field as follows:

i	A_i
0	01 A0 A1 A2 A3 A4 A5 A6 A7 03 02 01 00 06 00 00
1	01 A0 A1 A2 A3 A4 A5 A6 A7 03 02 01 00 06 00 01
2	01 A0 A1 A2 A3 A4 A5 A6 A7 03 02 01 00 06 00 02

12655

3. Parse the message *PlaintextData* as $M_1 || M_2$, where each message block M_i is a 16-octet string.

12656

4. The ciphertext blocks C_1, C_2 are computed as follows:

i	$AES(Key, A_i)$	$C_i = AES(Key, A_i) \oplus M_i$
1	12 5C A9 61 B7 61 6F 02 16 7A 21 66 70 89 F9 07	1A 55 A3 6A BB 6C 61 0D 06 6B 33 75 64 9C EF 10
2	CC 7F 54 D1 C4 49 B6 35 46 21 46 03 AA C6 2A 17	D4 66 4E CA D8 54 A8 35 46 21 46 03 AA C6 2A 17

12657

5. The string *Ciphertext* is the result of omitting all but the leftmost $l(m)=23$ octets of the string $C_1 || C_2$:

12658

$$CipherText = 1A 55 A3 6A BB 6C 61 0D 06 6B 33 75 64 9C EF 10 // D4 66 4E CA D8 54 A8$$

12660 6. Define the 16-octet encryption block S_0 by:

$$S_0 = E(Key, A_0) = B3\ 5E\ D5\ A6\ DC\ 43\ 6E\ 49\ D6\ 17\ 2F\ 54\ 77\ EB\ B4\ 39$$

12662 7. The encrypted authentication tag U is the result of XOR-ing the string consisting of the leftmost $M=8$ octets of S_0 and the authentication tag T :

$$U = 0A\ 89\ 5C\ C1\ D8\ FF\ 94\ 69$$

12665 **Output:** the right-concatenation c of the encrypted message *Ciphertext* and the encrypted authentication tag U :

$$c = 1A\ 55\ A3\ 6A\ BB\ 6C\ 61\ 0D\ 06\ 6B\ 33\ 75\ 64\ 9C\ EF\ 10\ //\ D4\ 66\ 4E\ CA\ D8\ 54\ A8\ //\ 0A\ 89\ 5C\ C1\ D8\ FF\ 94\ 69$$

12669 **C.4 CCM* Mode Decryption and Authentication 12670 Checking Transformation**

12671 This annex provides sample test vectors for the inverse of the mode of operation as specified in section B.1.2.

12672 **Prerequisites:** The following prerequisites are established for the operation of the mode of operation:

12673 1. The parameter M shall have the integer value 8.

12674 **Input:** The inputs to the inverse mode of operation are:

12675 1. The key *Key* of size *keylen*=128 bits to be used:

$$Key = C0\ C1\ C2\ C3\ C4\ C5\ C6\ C7\ C8\ C9\ CA\ CB\ CC\ CD\ CE\ CF$$

12677 2. The nonce *N* of 15- $L=13$ octets to be used:

$$Nonce = A0\ A1\ A2\ A3\ A4\ A5\ A6\ A7\ //\ 03\ 02\ 01\ 00\ //\ 06$$

12679 3. The octet string c of length $l(c)=31$ octets to be used:

$$c = 1A\ 55\ A3\ 6A\ BB\ 6C\ 61\ 0D\ 06\ 6B\ 33\ 75\ 64\ 9C\ EF\ 10\ //\ D4\ 66\ 4E\ CA\ D8\ 54\ A8\ //\ 0A\ 89\ 5C\ C1\ D8\ FF\ 94\ 69$$

12682 4. The octet string a of length $l(a)=8$ octets to be used:

$$a = 00\ 01\ 02\ 03\ 04\ 05\ 06\ 07$$

12684 **C.4.1 Decryption Transformation**

12685 The decryption transformation involves the following steps, in order:

12686 1. Parse the message c as $C \parallel U$, where the rightmost string U is an M -octet string:

$$C = 1A\ 55\ A3\ 6A\ BB\ 6C\ 61\ 0D\ 06\ 6B\ 33\ 75\ 64\ 9C\ EF\ 10\ //\ D4\ 66\ 4E\ CA\ D8\ 54\ A8;\
U = 0A\ 89\ 5C\ C1\ D8\ FF\ 94\ 69$$

12689 2. Form the padded message *CiphertextData* by right-concatenating the string C with the smallest non-negative number of all-zero octets such that the octet string *CiphertextData* has length divisible by 16.

$$CipherTextData = 1A\ 55\ A3\ 6A\ BB\ 6C\ 61\ 0D\ 06\ 6B\ 33\ 75\ 64\ 9C\ EF\ 10\ //\ D4\ 66\ 4E\ CA\ D8\ 54\ A8\ //\ 00\ 00\ 00\ 00\ 00\ 00\ 00\ 00$$

12694 3. Form the 1-octet *Flags* field as follows:

$$Flags = 01$$

12696

4. Define the 16-octet A_i field as follows:

i	A_i
0	01 A0 A1 A2 A3 A4 A5 A6 A7 03 02 01 00 06 00 00
1	01 A0 A1 A2 A3 A4 A5 A6 A7 03 02 01 00 06 00 01
2	01 A0 A1 A2 A3 A4 A5 A6 A7 03 02 01 00 06 00 02

12697

5. Parse the message $CiphertextData$ as $C_1 || C_2$, where each message block C_i is a 16-octet string.

12698

6. The ciphertext blocks P_1, P_2 are computed as follows.

I	$\text{AES}(\text{Key}, A_i)$	$P_i = \text{AES}(\text{Key}, A_i) \oplus C_i$
1	12 5C A9 61 B7 61 6F 02 16 7A 21 66 70 89 F9 07	08 09 0A 0B 0C 0D 0E 0F 10 11 12 13 14 15 16 17
2	CC 7F 54 D1 C4 49 B6 35 46 21 46 03 AA C6 2A 17	18 19 1A 1B 1C 1D 1E 00 00 00 00 00 00 00 00

12699

7. The octet string m is the result of omitting all but the leftmost $l(m)=23$ octets of the string $P_1 || P_2$:

12700

$$m = 08 09 0A 0B 0C 0D 0E 0F 10 11 12 13 14 15 16 17 || 18 19 1A 1B 1C 1D 1E$$

12701

8. Define the 16-octet encryption block S_0 by

12702

$$S_0 = E(\text{Key}, A_0) = B3 5E D5 A6 DC 43 6E 49 D6 17 2F 54 77 EB B4 39$$

12703

9. The purported authentication tag T is the result of XOR-ing the string consisting of the leftmost $M=8$ octets of S_0 and the octet string U :

12704

$$T = B9 D7 89 67 04 BC FA 20$$

12705

C.4.2 Authentication Checking Transformation

12706

The authentication checking transformation involves the following steps:

12707

1. Form the message $AuthData$ using the input transformation in Input Transformation, with the string a as inputs and the octet string m that was established in section 1.4.1 (step 7):

12708

$$\begin{aligned} AuthData = & 00 08^1 01 02 03 04 05 06 07 00 00 00 00 00 00 || \\ & 08 09 0A 0B 0C 0D 0E 0F 10 11 12 13 14 15 16 17 \\ & 18 19 1A 1B 1C 1D 1E 00 00 00 00 00 00 00 00 \end{aligned}$$

12709

2. Use the authentication transformation in section C.3.2, with the message $AuthData$ to compute the authentication tag $MACTag$ as input:

12710

$$MACTag = B9 D7 89 67 04 BC FA 20$$

12711

3. Compare the output tag $MACTag$ resulting from this transformation with the tag T that was established in section 4.1 (step 9):

12712

$$T = B9 D7 89 67 04 BC FA 20 = MACTag$$

¹ CCB 1520

12719 **Output:** Since $MACTag=T$, output ‘valid’ and accept the octet string m and accept one of the key sharing
12720 group member(s) as the source of m .

C.5 Cryptographic Hash Function

12722 This annex provides sample test vectors for the cryptographic hash function specified in clause B.1.3.

C.5.1 Test Vector Set 1

12724 **Input:** The input to the cryptographic hash function is as follows:

- 12725 1. The bit string M of length $l=8$ bits to be used:

12726 $M=C0$

12727 **Actions:** The hash value shall be derived as follows:

- 12728 1. Pad the message M by right-concatenating to M the bit ‘1’ followed by the smallest non-negative number
12729 of ‘0’ bits, such that the resulting string has length 14 (**mod** 16) octets:

12730 $C0 \parallel 80\ 00\ 00\ 00\ 00\ 00\ 00\ 00\ 00\ 00\ 00\ 00\ 00\ 00\ 00\ 00$

- 12731 2. Form the padded message M' by right-concatenating to the resulting string the 16-bit string that is equal
12732 to the binary representation of the integer l :

12733 $M' = C0 \parallel 80\ 00\ 00\ 00\ 00\ 00\ 00\ 00\ 00\ 00\ 00\ 00\ 00\ 00\ 00\ 08$

- 12734 3. Parse the padded message M' as M_i , where each message block M_i is a 16-octet string.

- 12735 4. The hash value $Hash_1$ is computed as follows:

i	Hash _i	M _i
0	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	∞
1	AE 3A 10 2A 28 D4 3E E0 D4 A0 9E 22 78 8B 20 6C	C0 80 00 00 00 00 00 00 00 00 00 00 00 00 00 08

12736 **Output:** the 16-octet string $Hash = Hash_1 = AE\ 3A\ 10\ 2A\ 28\ D4\ 3E\ E0\ D4\ A0\ 9E\ 22\ 78\ 8B\ 20\ 6C$.

C.5.2 Test Vector Set 2

12738 **Input:** The input to the cryptographic hash function is as follows:

- 12739 1. The bit string M of length $l=128$ bits to be used:

12740 $M=C0\ C1\ C2\ C3\ C4\ C5\ C6\ C7\ C8\ C9\ CA\ CB\ CC\ CD\ CE\ CF$

12741 **Actions:** The hash value shall be derived as follows:

- 12742 1. Pad the message M by right-concatenating to M the bit ‘1’ followed by the smallest non-negative number
12743 of ‘0’ bits, such that the resulting string has length 14 (**mod** 16) octets:

12744 $C0\ C1\ C2\ C3\ C4\ C5\ C6\ C7\ C8\ C9\ CA\ CB\ CC\ CD\ CE\ CF\ ||$
12745 $80\ 00\ 00\ 00\ 00\ 00\ 00\ 00\ 00\ 00\ 00\ 00\ 00\ 00\ 00\ 00$

- 12746 2. Form the padded message M' by right-concatenating to the resulting string the 16-bit string that is equal
12747 to the binary representation of the integer l :

12748 $M' = \text{C0 C1 C2 C3 C4 C5 C6 C7 C8 C9 CA CB CC CD CE CF} \parallel$
12749 $80 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 \parallel 00 80$

- 12750 3. Parse the padded message M' as $M_1 \parallel M_2$, where each message block M_i is a 16-octet string.

12751

4. The hash value $Hash_2$ is computed as follows:

i	Hash _i	M _i
0	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	∞
1	84 EE 75 E5 4F 9A 52 0F 0B 30 9C 35 29 1F 83 4F	C0 C1 C2 C3 C4 C5 C6 C7 C8 C9 CA CB CC CD CE CF
2	A7 97 7E 88 BC 0B 61 E8 21 08 27 10 9A 22 8F 2D	80 00 00 00 00 00 00 00 00 00 00 00 00 00 00 08

12752 **Output:** the 16-octet string $Hash = Hash_2 = A7 97 7E 88 BC 0B 61 E8 21 08 27 10 9A 22 8F 2D$.

12753

C.5.3 Test Vector Set 3

12754

Input: The input to the cryptographic hash function is as follows:

12755

1. The bit string M of length l = 65528 bits to be used.

12756

2. 8191 bytes (sequence of 0, 1, 2, ... 255, 0, 1, 2, ...)

12757

3. This test vector is beneath the threshold of a 216 bit string so the first padding method described in clause B.6 is utilized.

12759

Actions: The hash value shall be derived as follows:

12760

1. Pad the message by right-concatenating to M the bit 1 followed by the smallest non-negative number of '0' bits, such that the resulting string has length 14 (mod 16) octets:

12762

00 01 02 03 04 ... FB FC FD FE || 80 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00

12763

2. Form the padded message M' by right-concatenating to the resulting string the 16-bit string that is equal to the binary representation of the integer l:

12765

00 01 02 03 04 ... FB FC FD FE || 80 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 || FF F8

12766

3. Parse the padded message M' as M_i, where each message block M_i is a 16-octet string.

12767

4. The hash value Hash_i is computed as follows using 16-byte hash block operations:

i	Hash _i	M _i
0	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	-
1	7A CB 0D DA B8 D3 EA 7B 97 9E 4C 6D 1A EB AC 8D	00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F
...
i - 1	C3 22 D1 D3 9D 10 86 43 82 06 BD EB 26 41 66 1C	F0 F1 F2 F3 F4 F5 F6 F7 F8 F9 FA FB FC FD FE 80
i	24 EC 2F E7 5B BF FC B3 47 89 BC 06 10 E7 F1 65	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 FF F8

12768 C.5.4 Test Vector 4

12769 **Input:** The input to the cryptographic hash function is as follows:

- 12770 1. The bit string M of length $l = 65536$ bits to be used.
- 12771 2. 8192 bytes (sequence of 0, 1, 2, ... 255, 0, 1, 2, ...)
- 12772 3. This test vector is above the threshold of a 216 bit string so the second padding method described in
12773 clause B.6 is utilized.

12774 **Actions:** The hash value shall be derived as follows.

- 12775 1. Pad the message by right-concatenating to M the bit 1 followed by the smallest non-negative number of
12776 '0' bits, such that the resulting string has length $10 \pmod{16}$ octets:

12777 00 01 02 03 04 ... FB FC FD FE FF || 80 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00

- 12778 2. Form the padded message M' by right-concatenating to the resulting string the 32-bit string that is equal
12779 to the binary representation of the integer l :

12780 00 01 02 03 04 ... FB FC FD FE FF || 80 00 00 00 00 00 00 00 00 00 00 00 00 00 || 00 01 00 00

- 12781 3. Concatenate a 16-bit string of zeros for the padding normally used by the first padding method described
12782 in clause B.6.

12783 00 01 02 03 04 ... FB FC FD FE FF || 80 00 00 00 00 00 00 00 00 00 00 00 00 00 || 00 01 00 00 || 00 00

- 12784 4. Parse the padded message M' as M_i , where each message block M_i is a 16-octet string.

- 12785 5. The hash value $Hash_i$ is computed as follows using 16-byte hash block operations:

i	Hash _i	M _i
0	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	-
1	7A CB 0D DA B8 D3 EA 7B 97 9E 4C 6D 1A EB AC 8D	00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F
...
i - 1	4E 55 0D CE 34 31 42 96 41 BA D0 C7 BC 44 34 67	F0 F1 F2 F3 F4 F5 F6 F7 F8 F9 FA FB FC FD FE FF
i	DC 6B 06 87 F0 9F 86 07 13 1C 17 0B 3B D3 15 91 ²	80 00 00 00 00 00 00 00 00 00 00 00 00 01 00 00

12786 C.5.5 Test Vector 5

12787 **Input:** The input to the cryptographic hash function is as follows:

- 12788 1. The bit string M of length $l = 65608$ bits to be used.
- 12789 2. 8201 bytes (sequence of 0, 1, 2, ... 255, 0, 1, 2, ...)
- 12790 3. This test vector is above the threshold of a 216 bit string so the second padding method described in
12791 clause B.6 is utilized.

² CCB 1519

- 12792 **Actions:** The hash value shall be derived as follows.
- 12793 1. Pad the message by right-concatenating to M the bit 1 followed by the smallest non-negative number of '0' bits, such that the resulting string has length 10 (mod 16) octets:
12794 00 01 02 03 04 ... 04 05 06 07 08 || 80
- 12795 2. Form the padded message M' by right-concatenating to the resulting string the 32-bit string that is equal to the binary representation of the integer l :
12796 00 01 02 03 04 ... 04 05 06 07 08 || 80 || 00 01 00 48
- 12797 3. Concatenate a 16-bit string of zeros for the padding normally used by the first padding method described in clause B.6.
12798 00 01 02 03 04 ... 04 05 06 07 08 || 80 || 00 01 00 48 || 00 00
- 12799 4. Parse the padded message M' as M_i , where each message block M_i is a 16-octet string.
12800 5. The hash value $Hash_i$ is computed as follows using 16-byte hash block operations:

i	Hash _i	M _i
0	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	-
1	7A CB 0D DA B8 D3 EA 7B 97 9E 4C 6D 1A EB AC 8D	00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F
...
i - 1	4E 55 0D CE 34 31 42 96 41 BA D0 C7 BC 44 34 67	F0 F1 F2 F3 F4 F5 F6 F7 F8 F9 FA FB FC FD FE FF
i	72 C9 B1 5E 17 8A A8 43 E4 A1 6C 58 E3 36 43 A3	00 01 02 03 04 05 06 07 08 80 00 01 00 48 00 00

C.5.6 Test Vector 6

- 12804 **Input:** The input to the cryptographic hash function is as follows:
- 12805 1. The bit string M of length $l = 65616$ bits to be used.
- 12806 2. 8202 bytes (sequence of 0, 1, 2, ... 255, 0, 1, 2, ...)
- 12807 3. This test vector is above the threshold of a 216 bit string so the second padding method described in clause B.6 is utilized.
- 12808 **Actions:** The hash value shall be derived as follows.
- 12809 1. Pad the message by right-concatenating to M the bit 1 followed by the smallest non-negative number of '0' bits, such that the resulting string has length 10 (mod 16) octets:
12810 00 01 02 03 04 ... 05 06 07 08 09 || 80 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
- 12811 2. Form the padded message M' by right-concatenating to the resulting string the 32-bit string that is equal to the binary representation of the integer l :
12812 00 01 02 03 04 ... 05 06 07 08 09 || 80 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 || 00 01 00 50

12817 3. Concatenate a 16-bit string of zeros for the padding normally used by the first padding method described
12818 in clause B.6.

12819 00 01 02 03 04 ... 05 06 07 08 09 || 80 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 || 00 01 00 50 || 00 00

12820 4. Parse the padded message M' as M_1 , where each message block M_i is a 16-octet string.

12821 5. The hash value $Hash_1$ is computed as follows using 16-byte hash block operations:

i	Hash _i	M _i
0	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	-
1	7A CB 0D DA B8 D3 EA 7B 97 9E 4C 6D 1A EB AC 8D	00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F
...
i - 1	CC C1 F8 A3 D5 6A 93 20 41 08 10 2B 46 25 0D A7	00 01 02 03 04 05 06 07 08 09 80 00 00 00 00 00
i	BC 98 28 D5 9B 2A A3 23 DA F2 0B E5 F2 E6 65 11	00 00 00 00 00 00 00 00 00 00 00 00 00 00 01 00 50 00 00

C.6 Keyed Hash Function for Message Authentication

12823 This annex provides sample test vectors for the keyed hash function for message authentication as specified
12824 in clause B.1.4.

C.6.1 Test Vector Set 1

12826 **Input:** The input to the keyed hash function is as follows:

12827 1. The key Key of size $keylen=128$ bits to be used:

12828 $Key = 40\ 41\ 42\ 43\ 44\ 45\ 46\ 47\ 48\ 49\ 4A\ 4B\ 4C\ 4D\ 4E\ 4F$

12829 2. The bit string M of length $l=8$ bits to be used:

12830 $M=C0$

12831 **Actions:** The keyed hash value shall be derived as follows:

12832 1. Create the 16-octet string $ipad$ (inner pad) as follows:

12833 $ipad = 36\ 36\ 36\ 36\ 36\ 36\ 36\ 36\ 36\ 36\ 36\ 36\ 36\ 36\ 36\ 36$

12834 2. Form the inner key Key_1 by XOR-ing the bit string Key and the octet string $ipad$:

12835 $Key_1 = Key \oplus ipad = 76\ 77\ 74\ 75\ 72\ 73\ 70\ 71\ 7E\ 7F\ 7C\ 7D\ 7A\ 7B\ 78\ 79$

12836 3. Form the padded message M_1 by right-concatenating the bit string Key_1 with the bit string M :

12837 $M_1 = Key_1 \parallel M = 76\ 77\ 74\ 75\ 72\ 73\ 70\ 71\ 7E\ 7F\ 7C\ 7D\ 7A\ 7B\ 78\ 79 \parallel C0$

- 12838 4. Compute the hash value $Hash_1$ of the bit string M_1 :
12839 $Hash_1 = 3C\ 3D\ 53\ 75\ 29\ A7\ A9\ A0\ 3F\ 66\ 9D\ CD\ 88\ 6C\ B5\ 2C$
12840 5. Create the 16-octet string $opad$ (outer pad) as follows:
12841 $opad = 5C\ 5C$
12842 6. Form the outer key Key_2 by XOR-ing the bit string Key and the octet string $opad$:
12843 $Key_2 = Key \oplus opad = 1C\ 1D\ 1E\ 1F\ 18\ 19\ 1A\ 1B\ 14\ 15\ 16\ 17\ 10\ 11\ 12\ 13$
12844 7. Form the padded message M_2 by right-concatenating the bit string Key_2 with the bit string $Hash_1$:
12845 $M_2 = Key_2 \parallel Hash_1 = 1C\ 1D\ 1E\ 1F\ 18\ 19\ 1A\ 1B\ 14\ 15\ 16\ 17\ 10\ 11\ 12\ 13 \parallel$
12846 $3C\ 3D\ 53\ 75\ 29\ A7\ A9\ A0\ 3F\ 66\ 9D\ CD\ 88\ 6C\ B5\ 2C$
12847 8. Compute the hash value $Hash_2$ of the bit string M_2 :
12848 $Hash_2 = 45\ 12\ 80\ 7B\ F9\ 4C\ B3\ 40\ 0F\ 0E\ 2C\ 25\ FB\ 76\ E9\ 99$
12849 **Output:** the 16-octet string $HMAC = Hash_2 = 45\ 12\ 80\ 7B\ F9\ 4C\ B3\ 40\ 0F\ 0E\ 2C\ 25\ FB\ 76\ E9\ 99$

12850 C.6.2 Test Vector Set 2

12851 **Input:** The input to the keyed hash function is as follows:

- 12852 1. The key Key of size $keylen=256$ bits to be used:
12853 $Key = 40\ 41\ 42\ 43\ 44\ 45\ 46\ 47\ 48\ 49\ 4A\ 4B\ 4C\ 4D\ 4E\ 4F \parallel$
12854 $50\ 51\ 52\ 53\ 54\ 55\ 56\ 57\ 58\ 59\ 5A\ 5B\ 5C\ 5D\ 5E\ 5F$
12855 2. The bit string M of length $l=128$ bits to be used:
12856 $M = C0\ C1\ C2\ C3\ C4\ C5\ C6\ C7\ C8\ C9\ CA\ CB\ CC\ CD\ CE\ CF$

12857 **Actions:** The keyed hash value shall be derived as follows:

- 12858 1. Compute the hash value Key_0 of the bit string Key :
12859 $Key_0 = 22\ F4\ 0C\ BE\ 15\ 66\ AC\ CF\ EB\ 77\ 77\ E1\ C4\ A9\ BB\ 43$
12860 2. Create the 16-octet string $ipad$ (inner pad) as follows:
12861 $ipad = 36\ 36\ 36\ 36\ 36\ 36\ 36\ 36\ 36\ 36\ 36\ 36\ 36\ 36\ 36\ 36$
12862 3. Form the inner key Key_1 by XOR-ing the bit key Key_0 and the octet string $ipad$:
12863 $Key_1 = Key_0 \oplus ipad = 14\ C2\ 3A\ 88\ 23\ 50\ 9A\ F9\ DD\ 41\ 41\ D7\ F2\ 9F\ 8D\ 75$
12864 4. Form the padded message M_1 by right-concatenating the bit string Key_1 with the bit string M :
12865 $M_1 = Key_1 \parallel M = 14\ C2\ 3A\ 88\ 23\ 50\ 9A\ F9\ DD\ 41\ 41\ D7\ F2\ 9F\ 8D\ 75 \parallel$
12866 $C0\ C1\ C2\ C3\ C4\ C5\ C6\ C7\ C8\ C9\ CA\ CB\ CC\ CD\ CE\ CF$
12867 5. Compute the hash value $Hash_1$ of the bit string M_1 :
12868 $Hash_1 = 42\ 65\ BE\ 29\ 74\ 55\ 8C\ A2\ 7B\ 77\ 85\ AC\ 73\ F2\ 22\ 10$
12869 6. Create the 16-octet string $opad$ (outer pad) as follows:
12870 $opad = 5C\ 5C$
12871 7. Form the outer key Key_2 by XOR-ing the bit string Key_0 and the octet string $opad$:
12872 $Key_2 = Key_0 \oplus opad = 7E\ A8\ 50\ E2\ 49\ 3A\ F0\ 93\ B7\ 2B\ 2B\ BD\ 98\ F5\ E7\ 1F$

- 12873 8. Form the padded message M_2 by right-concatenating the bit string Key_2 with the bit string $Hash_1$:
- 12874 $M_2 = Key_2 \parallel Hash_1 = 7E\ A8\ 50\ E2\ 49\ 3A\ F0\ 93\ B7\ 2B\ 2B\ BD\ 98\ F5\ E7\ 1F \parallel$
- 12875 42\ 65\ BE\ 29\ 74\ 55\ 8C\ A2\ 7B\ 77\ 85\ AC\ 73\ F2\ 22\ 10
- 12876 9. Compute the hash value $Hash_2$ of the bit string M_2 :
- 12877 $Hash_2 = A3\ B0\ 07\ 99\ 84\ BF\ 15\ 57\ F7\ 4A\ 0D\ 63\ 87\ E0\ A1\ 1A$
- 12878 **Output:** the 16-octet string $HMAC = Hash_2 = A3\ B0\ 07\ 99\ 84\ BF\ 15\ 57\ F7\ 4A\ 0D\ 63\ 87\ E0\ A1\ 1A$

12879 **C.6.3 Specialized Keyed Hash Function for Message** 12880 **Authentication**

12881 This annex provides sample test vectors for the specialized keyed hash function for message authentication as
12882 specified in clause B.1.4.

12883 For test vectors, see clause C.6.

12884 10.

12886

12887

12888

12889

12890

12891

12892

12893

12894

12895

12896

12897

12898

12899

12900

12901

12902

12903

12904

12905

This page intentionally left blank.

ANNEC D **MAC AND PHY SUB-LAYER CLARIFICATIONS**

D.1 Introduction

D.1.1 Scope

This annex applies to the IEEE 802.15.4 2003 Medium Access Control sub-layer (MAC) and Physical Layer (PHY) specification when used in conjunction with higher layers defined by the ZigBee specification. Nothing is implied about the usage under other circumstances.

D.1.2 Purpose

The current ZigBee specification assumes the use of the MAC and PHY sub-layers defined in the IEEE 802.15.4 2003 specification. However, as developers have put the MAC and PHY sub-layers into use, they have uncovered problems that may or may not have been anticipated by the authors of the specification, or are not covered in the IEEE 802.15.4 2003 specification. This document is intended to provide solutions to such problems, for use by the ZigBee Alliance.

D.2 Stack Size Issues

Both MAC and ZigBee stack developers have discovered that implementation of a full-blown MAC is a major undertaking and requires a great deal of code space. Even with the optional GTS and MAC security features eliminated, it is not surprising to find the MAC taking up more than 24K of code space on a processor with 64K of available space.

The ZigBee Alliance has adopted a compensating policy to declare MAC features that are not required to support a particular stack profile optional with respect to that stack profile. In particular, any MAC feature that will not be exploited as a result of platform compliance testing for a particular stack profile need not be present in order for an implementation to be declared platform compliant. For example, since the ZigBee Pro stack profile relies on a beaconless network, the platform compliance testing for the stack profile does not employ beaconing. The code to support regular beaconing, beacon track, and so on, may therefore be absent from the code base of the device under test without the knowledge of the testers, without presenting a problem with respect to platform compliance certification.

The exact list of MAC features that must be supported in a platform is described in the PICS document used for MAC conformance testing.

12934

D.3 MAC Association

12935
12936
12937
12938
12939

At association time, according to the IEEE 802.15.4 specification, a number of frames are sent, including an association request command, an associate response command and a data request. There is some ambiguity in the specification regarding the addressing fields in the headers for these frames. Table D.1 to Table D.3 outline the allowable options that shall be recognized by devices implementing the ZigBee specification. In each case, the first option given is the preferred option and should be used.

12940

Table D.1 Associate Request Header Fields

DstPANId	DstAddr	SrcPANId	SrcAddr
The PANId of the destination device.	The 16-bit short address of the destination device.	0xffff	The 64-bit extended address of the source device.
		PANId omitted because the IntraPAN sub-field in the frame control field is set to one.	
		The PANId of the destination device.	
Not present if the destination device is the PAN coordinator.	Not present if the destination device is the PAN coordinator.		

12941

12942

Note that in this case and the case below, the source of the command is the device requesting association.

12943

Table D.2 Data Request Header Fields

DstPANId	DstAddr	SrcPANId	SrcAddr
The PANId of the destination device.	The 16-bit short address of the destination device.	0xffff	The 64-bit extended address of the source device.
		PANId omitted because the IntraPAN sub-field in the frame control field is set to one.	
		The PANId of the destination device.	
Not present if the destination device is the PAN coordinator.	Not present if the destination device is the PAN coordinator.		

12944

Table D.3 Association Response Header Fields

DstPANId	DstAddr	SrcPANId	SrcAddr
The PANId of the destination device.	The 64-bit extended address of the destination device.	PANId omitted because the IntraPAN sub-field in the frame control field is set to one.	The 64-bit extended address of the source device.
		The PANId of the source device.	
0xffff			

12945

D.4 aMaxMACFrameSize

12946
12947
12948
12949

The IEEE 802.15.4 MAC specification [B1] has two constants that define the minimum and maximum values for the MAC data packet payload size. These are the *aMaxMACPayloadSize* (118 bytes) and the *aMaxMACSafePayloadSize* (102 bytes). Since the overhead imposed by the MAC header is variable, the actual limit of the MAC data payload size is in between these values and may vary by implementation.

12950
12951
12952
12953

When used in a ZigBee platform, the MAC implementation must support transmission and reception of unsecured MAC data packet payloads of up to (*aMaxPHYPacketSize* - *nwkMinHeaderOverhead*) bytes. The value of *nwkMinHeaderOverhead* parameter takes into account the fact that ZigBee uses short addressing modes and intra-PAN communications.

12954

D.5 Frame Version Value

12955
12956
12957
12958
12959
12960
12961
12962

The MAC specification requires that any unsecured MAC data packet with payload size greater than *aMaxMACSafePayloadSize* (102bytes) must have the Frame Version field set to one (see section 6.3.1 of [B1]). When used in a ZigBee platform, the MAC implementation must always set the Frame Version field in unsecured MAC data packets to zero. The reason for this is to ensure backwards compatibility with existing deployed ZigBee devices that cannot receive packets correctly if these bits are set to a non-zero value. Note that this deviation is only on the transmit side, the receive side processing is unchanged. That is, the MAC implementation must be able to receive and process MAC data packets with the Frame Version field set to any non-reserved value, as specified in section 5.6.1.2 of [B1].

12963

The MAC specification allows the coordinator realignment command to be sent with either Frame Version of zero or one. The format of the command is different in each case (see section 5.3.8.1 of [B1]). When used in a ZigBee implementation, the MAC implementation must always set the Frame Version field in the coordinator realignment command to zero.

12968

12969 **D.6 CSMA Backoff Timing**

12970 The IEEE 802.15.4 2006 specification provides an increase in macMaxBE to 8 from 5. This higher value is
12971 allowed within ZigBee and it is recommended as the default. The default value of macMinBE should be 5
12972 instead of 3. This provides better joining performance in dense networks where many devices may be re-
12973 sponding to a beacon request.

12974 Note the time a device listens for beacons is set by IEEE 802.15.4 to aBaseSuperframeDuration*(2n+1)
12975 symbols where n is the value of the *ScanDuration* parameter. For ZigBee implementations the value of n
12976 should be set to ensure the duration of the listening window is similar to the length of time the beacon re-
12977 sponds are expected.

12978

12979 **D.7 MAC Interface Changes**

12980 The IEEE-802-15-4 specification has no notification when a MAC data poll is received by a coordinator
12981 (FFD) or any ability for the ZigBee layers to dictate the response to the MAC data poll. Therefore the fol-
12982 lowing interfaces are defined for a MAC used by ZigBee network layers.

12983 **D.7.1 Additional Primitives accessed through the** 12984 **MLME-SAP**

12985 Those primitives marked with a diamond (◊) are optional for an RFD.

Name	Request	Indication	Response	Confirm
MLME-Poll	(Already specified in reference B1)	D.7.2 ◊	-	(Already specified in reference B1)

12986 **D.7.2 MLME-POLL.indication**

12987 The MLME-Poll.indication primitive notifies the next higher level that a request for data has been received.

12988 **D.7.2.1 Semantics of the service primitive**

12989 The semantics of the MLME-Poll.indication primitive is as follows.

12990 MLME-Poll.indication (

12991 AddrMode

12992 DeviceAddress

12993)

Name	Type	Valid Range	Description
AddrMode	Integer	0x02 – 0x03	This value can take one of the following values: 2=16 bit short address. 3=64 bit extended address.
DeviceAddress	Integer	As specified by Ad-	The address of the device requesting pending

		drMode parameter.	data.
--	--	-------------------	-------

12995

12996 **D.7.2.2 When Generated**

12997 The MLME-POLL.indication primitive indicates the reception of a Data request command frame by the
12998 MAC sub-layer and issued to the local SSCS (service specific convergence sublayer).

12999 **D.7.2.3 Effect on Receipt**

13000 The effect on receipt of the MLME-Poll.indication primitive is that the next higher layer is notified that a
13001 device is requesting to see if there is a pending MAC data frame. If an indirect frame is queued by the
13002 higher layer during the processing of an MLME-POLL.indication it shall affect the pending bit in the ACK
13003 frame corresponding to the data request frame that caused the MLME-POLL.indication to be issued.

13004

13005

13006

13007

13008

13009

13010

13011

13012

13013

13014

13015

13016

13017

13018

13019

13020

13021

13022

13023

13024

13025

13026

This page intentionally left blank.

13027 **ANNEX E OPERATING NETWORK**

13028 **MANAGER AS NETWORK CHANNEL**

13029 **MANAGER FOR INTERFERENCE**

13030 **REPORTING AND RESOLUTION**

13031 **Prerequisites:** Devices shall limit their operations to channels within their current PHY (i.e.
13032 868/915 MHz or 2450 MHz). Commands including channels outside the band shall be ignored.

13033 A single device can become the Network Channel Manager. This device acts as the central mechanism
13034 for reception of network interference reports and changing the channel of the network if interference
13035 is detected. The default address of the network manager is the coordinator, however this
13036 can be updated by sending a Mgmt_NWK_Update_req command with a different short address for
13037 the network channel manager. The device that is the Network Channel Manager shall set the net-
13038 work manager bit in the server mask in the node descriptor and shall respond to Sys-
13039 tem_Server_Discovery_req commands.

13040 Each router or coordinator is responsible for tracking transmit failures using the TransmitFailure
13041 field in the neighbor table and also keeping a NIB counter for total transmissions attempted. A de-
13042 vice that detects a significant number of transmission failures may take action to determine if inter-
13043 ference is a cause. The following steps are an example of that procedure¹:

1. Conduct an energy scan on all channels within the current PHY. If this energy scan does not indicate higher energy on the current channel than other channels, no action is taken. The device should continue to operate as normal and the message counters are not reset. However, repeated energy scans are not desirable as the device is off the network during these scans and therefore implementations should limit how often a device with failures conducts energy scans.
2. If the energy scan does indicate increased energy on the channel in use, a Mgmt_NWK_Update_notify should be sent to the Network Manager to indicate interference is present. This report is sent as an APS Unicast with acknowledgement and once the acknowledgement is received the total transmit and transmit failure counters are reset to zero.
3. To avoid a device with communication problems from constantly sending reports to the network manager, the device should not send a Mgmt_NWK_Update_notify more than 4 times per hour.

13056 Upon receipt of an unsolicited Mgmt_NWK_Update_notify, the network manager must evaluate if
13057 a channel change is required in the network. The specific mechanisms the network manager uses to
13058 decide upon a channel change are left to the implementers. It is expected that implementers will
13059 apply different methods to best determine when a channel change is required and how to select the
13060 most appropriate channel. The following is offered as guidance for implementation.

¹ CCB 1493

- 13061 The network manager may do the following:
- 13062 1. Wait and evaluate if other reports from other devices are received. This may be appropriate if
13063 there are no other failures reported. In this case the network manager should add the reporting
13064 device to a list of devices that have reported interference. The number of devices on such a list
13065 would depend on the size of the network. The network manager can age devices out of this list.
- 13066 2. Request other interference reports using the Mgmt_NWK_Update_req command. This may be
13067 done if other failures have been reported or the network manager device itself has failures and a
13068 channel change may be desired. The network manager may request data from the list of devices
13069 that have reported interference plus other randomly selected routers in the network. The net-
13070 work manager should not request an update from the device that has just reported interference
13071 since this data is fresh already.
- 13072 3. Upon receipt of the Mgmt_NWK_Update_notify, the network manager shall determine if a
13073 channel change is required using whatever implementation specific mechanisms are considered
13074 appropriate. The network manager device with just one channel allowed in the *apsChannel-
13075 Mask* parameter must not issue the Mgmt_Nwk_Update_Req command to request other de-
13076 vices to change the current channel. However, the network manager may report channel quality
13077 issues to the application.
- 13078 4. If the above data indicate a channel change should be considered, the network manager com-
13079 pleted the following:
- 13080 a. Select a single channel based on the Mgmt_NWK_Update_notify based on the lowest
13081 energy. This is the proposed new channel. If this new channel does not have an energy
13082 level below an acceptable threshold, a channel change should not be done. Additionally, a
13083 new channel shall not belong to a PHY different from the one on which a network manager
13084 is operating now.
- 13085 5. Prior to changing channels, the network manager should store the energy scan value as the last
13086 energy scan value and the failure rate from the existing channel as the last failure rate. These
13087 values are useful to allow comparison of the failure rate and energy level on the previous
13088 channel to evaluate if the network is causing its own interference.
- 13089 6. The network manager should broadcast a Mgmt_NWK_Update_req notifying devices of the
13090 new channel. The broadcast shall be to all devices with RxOnWhenIdle equal to TRUE. The
13091 network manager is responsible for incrementing the *nwkUpdateId* parameter from the NIB and
13092 including it in the Mgmt_NWK_Update_req. The network manager shall set a timer based on
13093 the value of
13094 *apsChannelTimer* upon issue of a Mgmt_NWK_Update_req that changes channels and shall
13095 not issue another such command until this timer expires. However, during this period, the
13096 network manager can complete the above analysis. However, instead of changing channels, the
13097 network manager would report to the local application using Mgmt_NWK_Update_notify and
13098 the application can force a channel change using the Mgmt_NWK_Update_req.
- 13099 Upon receipt of a Mgmt_NWK_Update_req with a change of channels, the local network manager
13100 shall set a timer equal to the *nwkNetworkBroadcastDeliveryTime* and shall switch channels upon
13101 expiration of this timer. Each node shall also increment the *nwkUpdateId* parameter and also reset
13102 the total transmit count and the transmit failure counters.
- 13103 For devices with RxOnWhenIdle equals FALSE, any network channel change will not be received.
13104 On these devices or routers that have lost the network, an active scan shall be conducted on the
13105 *apsChannelMask* list in the APS IB using the extended PANID to find the network. If the extended
13106 PANID is found on different channels, the device should select the channel with the higher value in
13107 the *nwkUpdateId* parameter. If the extended PANID is not found using the *apsChannelMask* list, a
13108 scan should be completed using all channels within the current PHY.
- 13109

13110 ANNEX F

13111 USAGE OF MULTIPLE FRE- QUENCY BANDS

13112 F.1 Introduction

13113 F.1.1 Scope

13114 This annex clarifies uncertainties arising with ZigBee compliant devices that support several frequency
13115 bands.

13116 F.1.2 Purpose

13117 The ZigBee specification is based on the IEEE 802.15.4 ([B1]) standard that defines multiple PHYs. A
13118 compliant device shall support at least one of the following options: O-QPSK PHY at 2.4 GHz frequency
13119 band or the BPSK PHY at both 868 MHz and 915 MHz bands. Each of the frequency bands incorporates its
13120 own set of channels through a combination of channel numbers and channel pages. A ZigBee device shall use
13121 channel page zero which consists of the following channel numbers: channel 0 for the 868 MHz band,
13122 channels from 1 to 10 for the 915 MHz band and channels from 11 to 26 for the 2450 MHz band. Additionally
13123 the following apply:

- 13124 • A Zigbee compliant device declaring support of a frequency band shall support all the channels listed in
13125 channel page zero within that frequency band.
- 13126 • A Zigbee compliant device declaring support of the 868/915 MHz PHY shall support both 868 MHz and
13127 915 MHz frequency bands within this PHY

13128 F.2 Channels and Channel Masks Management General Guideline

13131 F.2.1 Channel Selection During Network Establishment

13132 When there is a set of devices intended to be a part of the same ZigBee network, with devices of that set,
13133 potentially, supporting different frequency bands, the coordinator, during network establishment, may
13134 choose a channel from a frequency band that is not supported by some of the other devices.

13135 Since, before a network is established, there is no mechanism for the coordinator to dynamically collect in-
13136 formation about frequency bands supported on each and every device in the network, this issue may be
13137 categorized as a network commissioning issue and has to be resolved in the layers above the ZigBee stack's
13138 core.

13139 **F.2.2 The Frequency Agility Feature Related Points**

13140 How a network manager or a device shall behave, considering the ability to support different frequency
13141 bands, is described in Annex E and in section 2.4.3.3.9.2. Implementers of the frequency agility feature
13142 should take into account that it is prohibited for a network manager device to move a network from one PHY
13143 to another. This limitation is introduced in order to avoid the situations when a part of devices in the network
13144 cannot physically migrate to a channel from another PHY and therefore got lost. At the same time moving a
13145 network from one frequency band to another within 868/915 MHz PHY is allowed since support of both
13146 bands is mandatory in accordance with IEEE P802.15.4 (§C.7.2.3 [B1]). The application layer must meet
13147 regional regulatory requirements by setting an appropriate value to the *apsChannelMask* parameter.

13148 **F.2.3 Network Management Services and Client Ser-** 13149 **vices Affected by Multiple Frequency Bands** 13150 **Support**

13151 The following Network Management Client Services and Network Management Services use the
13152 *ScanChannels* parameter and, therefore, have to be mentioned in regard of multiple frequency bands support:
13153 Mgmt_NWK_Disc_req, Mgmt_NWK_Update_req and NLME-JOIN.request. In case the *ScanChannels*
13154 bitmask includes a channel(s) from unsupported frequency band the INVALID_PARAMETER (see [B1])
13155 error status is supposed to be raised from the MAC layer to the NWK layer. If the destination addressing
13156 mode in the Mgmt_NWK_Disc_req and Mgmt_NWK_Update_req commands was unicast then the Remote
13157 Device shall incorporate the error status into the status field of the correspondent Mgmt_NWK_Disc_rsp and
13158 Mgmt_NWK_Update_rsp commands. The same error status shall be reported in NLME-JOIN.confirm
13159 primitive sent in response to an NLME-JOIN.request primitive if the latter contains unsupported channels.

13160 In case the NLME-JOIN.request primitive is used by the application layer to request a device to switch to a
13161 new channel (the *RejoinNetwork* parameter is equal to 0x03) then the application layer, by implemen-
13162 tation-specific means, has to ensure that the chosen channel is supported by all other devices in the network, to
13163 avoid the situation when some of the devices might be lost from the network due to inability to switch to an
13164 unsupported channel.

13165 **F.3 Timing Issues**

13166 Different frequency bands declared in the IEEE 802.15.4 2003 standard provide different bit rates. Therefore
13167 the ZigBee stack's time-related parameters have to be adjusted accordingly to achieve the stable operation on
13168 each of the supported frequency bands. The ZigBee stack's time-related parameters can be divided in two
13169 groups in regard of multiple frequency bands support: the first group includes time-related parameters that
13170 have a direct impact on the ZigBee stack's core's functioning and that ensure that the core's functioning is
13171 correct; the second group consists of the time-related parameters that have to be configured by an application.
13172 The ZigBee specification controls the first group of parameters and declares them in a way that makes them
13173 dependent on the currently used frequency band. These parameters are presented in Table F.1 and their values
13174 must be updated automatically each time a device migrates from one frequency band to another.

13175 **Table F.1 Internal Time-related Parameters**

Parameter	Reference
<i>:Config_NWK_Time_btwn_Scans</i>	Section 2.5.6.1, Table 2.149
<i>nwkcWaitBeforeValidation</i>	Section 3.5.1, Table 3.43

<i>nwkRouteDiscoveryTime</i>	Section 3.5.1, Table 3.43
<i>nwkMaxBroadcastJitter</i>	Section 3.5.1, Table 3.43
<i>nwkRREQRetryInterval</i>	Section 3.5.1, Table 3.43
<i>nwkMinRREQJitter</i>	Section 3.5.1, Table 3.43
<i>nwkMaxRREQJitter</i>	Section 3.5.1, Table 3.43
<i>nwkPassiveAckTimeout</i>	Section 3.5.2, Table 3.44
<i>nwkNetworkBroadcastDeliveryTime</i>	Section 3.5.2, Table 3.44
<i>apsSecurityTimeOutPeriod</i>	Section 4.4.10 Table 4.38

13176

13177

13178

13179

13180

13181

13182

13183

13184

13185

13186

13187

13188

13189

13190

13191

13192

13193

13194

13195

This page intentionally left blank.

13196

13197

ANNEX G

INTER-PAN COMMUNICATIONS

13198

13199

G.1 Scope and Purpose

13200

This annex defines a mechanism whereby ZigBee devices can perform exchanges of information with devices in their local area without having to form or join the same ZigBee network. This capability is used in a number of ZigBee functions from extending Smart Energy networks to simple low cost devices, for Green Power end devices, or for Touchlink commissioning.

13204

G.2 General Description

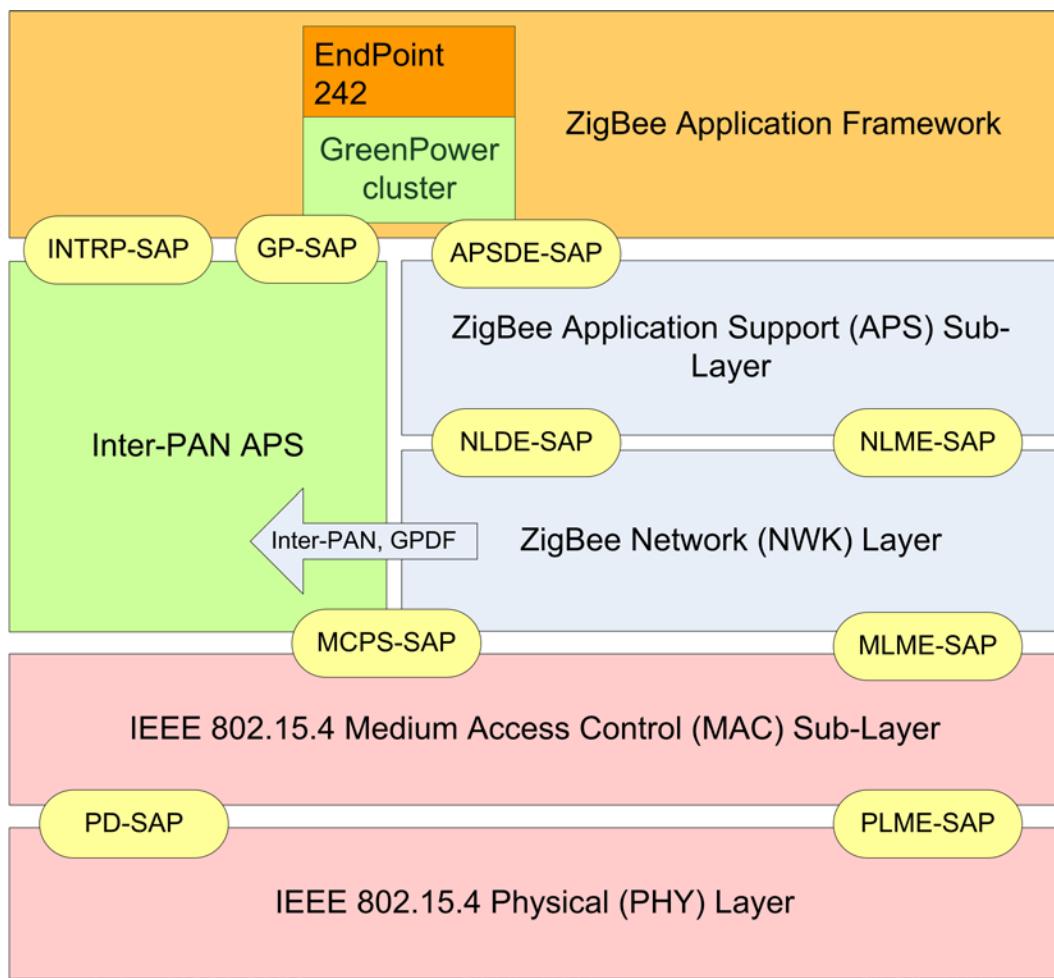
13205

G.2.1 What Inter-PAN APS Does

13206

A schematic view of the ZigBee stack enabling Inter-PAN data and Green Power Device Frame exchange is shown in Figure G.1.

13208

Figure G.1 ZigBee Stack with Inter-PAN APS

13209

13210

13211 Inter-PAN data exchanges and Green Power Device Frame (GPDF) exchanges are handled by a special “stub”
 13212 of the Application Support Sub-Layer, which is accessible through a special Service Access Point (SAP), the
 13213 INTRP-SAP, parallel to the APSDE-SAP. The Inter-PAN data exchange architecture is used by several
 13214 different mechanisms within ZigBee devices.

13215 The same Inter-PAN APS is intended to be used for these different services even if how they use it varies
 13216 slightly. In case of Inter-PAN data exchanges, the Inter-PAN APS performs just enough processing to pass
 13217 application data frames to the MAC for transmission and to pass Inter-PAN application frames from the
 13218 MAC to the application on receipt. In case of Green Power Device Frame exchanges, the Inter-PAN APS also
 13219 performs security processing of incoming and outgoing GPDF (see section G.5), as well as buffering of
 13220 outgoing GPDF (see section G.4.3). The incoming GPDF are delivered to the application on endpoint 242
 13221 and handled by that; see the specification of the Green Power cluster residing on endpoint 242 [B4].

13222 The use of Inter-PAN frames and Green Power Device Frames is indicated by the sub-fields of the network
 13223 Frame Control field, as described in section G.3.2.

13224 **G.2.2 Service Specification**

13225 The INTRP-SAP is a data service comprising eight primitives.

- INTRP-DATA.request - Provides a mechanism for a sending device to request transmission of an Inter-PAN message.
 - GP-DATA.request – Provides a mechanism for a sending device to request transmission of a Green Power Device Frame.
 - INTRP-DATA.confirm - Provides a mechanism for a sending device to understand the status of a previous request to send an Inter-PAN message.
 - GP-DATA.confirm - Provides a mechanism for a sending device to understand the status of a previous request to send a Green Power Device Frame.
 - INTRP-DATA.indication - Provides a mechanism for identifying and conveying an Inter-PAN message received from a sending device.
 - GP-DATA.indication - Provides a mechanism for identifying and conveying a Green Power Device Frame message received from a sending device.
 - GP-SEC.request – provides a mechanism for the Green Power Device Frame processing part of Inter-PAN APS to request security data from the Green Power application.
 - GP-SEC.response – provides a mechanism for the Green Power application to provide security data into the Green Power Device Frame processing part of the Inter-PAN APS.

G.2.3 The INTRP-DATA.request Primitive

The INTRP-DATA.request primitive allows an application entity to request data transmission via the Inter-PAN APS.

G.2.3.1 Semantics of the Service Primitive

```
INTRP-DATA.request {  
    SrcAddrMode  
    DstAddrMode  
    DstPANId  
    DstAddress  
    ProfileId  
    ClusterId  
    ASDULength  
    ASDU  
    ASDUHandle  
}
```

Table G.1 specifies the parameters of the INTRP-DATA.request primitive.

Table G.1 Semantics of the INTRP-DATA.request Primitive

Name	Type	Valid Range	Description
------	------	-------------	-------------

Name	Type	Valid Range	Description
SrcAddrMode	Integer	0x00 – 0x03	The source addressing mode for the MPDU to be sent. This value can take one of the following values: 0 x 00 = no address (SrcPANId and SrcAddress omitted). 0 x 01 = reserved. 0 x 02 = 16 bit short address. 0 x 03 = 64 bit extended address.
DstAddrMode	Integer	0x01 – 0x03	The addressing mode for the destination address used in this primitive. This parameter can take one of the values from the following list: 0x01 = 16-bit group address 0x02 = 16-bit NWK address, usually the broadcast address 0xffff 0x03 = 64-bit extended address
DstPANID	16-bit PAN ID	0x0000 – 0xFFFF	The 16-bit PAN identifier of the entity or entities to which the ASDU is being transferred or the broadcast PANId 0xffff.
DstAddress	16-bit or 64-bit address	As specified by the AddrMode parameter	The address of the entity or entities to which the ASDU is being transferred.
ProfileId	Integer	0x0000 – 0xffff	The identifier of the application profile for which this frame is intended.
ClusterId	Integer	0x0000 – 0xffff	The identifier of the cluster, within the profile specified by the ProfileId parameter, which defines the application semantics of the ASDU.
ASDULength	Integer	0x00 – (<i>aMaxMACFrameSize</i> - 9)	The number of octets in the ASDU to be transmitted.
ASDU	Set of octets	-	The set of octets forming the ASDU to be transmitted.
ASDUHandle	Integer	0x00 – 0xff	An integer handle associated with the ASDU to be transmitted.

13262 **G.2.3.2 When Generated**

13263 This primitive is generated by the local application entity when it wishes to address a frame to one or more
13264 peer application entities residing on neighboring devices using Inter-PAN data.

13265

13266 **G.2.3.3 Effect on Receipt**

13267 On receipt of the INTRP-DATA.request primitive by the Inter-PAN APS, the Inter-PAN APS will construct
13268 and transmit an Inter-PAN frame. This frame shall have a Protocol Version sub-field and the Frame
13269 Type sub-field of the NWK Frame Control field set to the values as specified in section G.3.2.1. The frame
13270 shall contain the given ASDU and set the parameters using the MCPS-DATA.request primitive of the MAC
13271 sub-layer, as described in section G.3.1.1. Once the corresponding MCPS-DATA.confirm primitive is re-
13272 ceived, the stack shall generate the INTRP-DATA.confirm primitive with a status value reflecting the status
13273 value returned by the MAC.

13274

G.2.4 The GP-DATA.request Primitive

13275 The GP-DATA.request primitive allows an application entity to request a Green Power data transmission via
13277 the Inter-PAN APS.

13278 **G.2.4.1 Semantics of the Service Primitive**

13279 The primitive interface is as follows:

13280 GP-DATA.request {
13281 Actions
13282 TxOptions
13283 ApplicationID
13284 SrcID
13285 GPD IEEE Address
13286 GPD CommandID
13287 ASDULength
13288 ASDU
13289 ASDUHandle
13290 gpTxQueueEntryLifetime
13291 }

13292 Table G.2 specifies the parameters of the GP-DATA.request primitive.

13293 **Table G.2 Parameters of the GP-DATA.request primitive**

Name	Type	Valid Range	Description
------	------	-------------	-------------

Name	Type	Valid Range	Description
Actions	Boolean	TRUE/FALSE	TRUE: add Green Power Device Frame into gpTxQueue FALSE: remove Green Power Device Frame from gpTxQueue
TxOptions	8-bit bitmap	Any valid	This provides transmission options for a Green Power Device Frame. There are a bitwise OR of one or more of the following: b0 = Use gpTxQueue b1 = use CSMA/CA b2 = use MAC ACK b3-b4 – Frame type for Tx (see values in Table G.10) b5 – b7 - reserved
ApplicationID	8 bit enumeration	0x00, 0x02	ApplicationID of the Green Power Device to which the frame will be sent. ApplicationID 0x00 indicates the usage of the 32 bit SrcID and ApplicationID 0x02 indicates the usage of the GPD IEEE address.
SrcID	Unsigned 32-bit Integer	0x00000000 – 0xffffffff	The identifier of the GPD entity to which the ASDU will be sent if ApplicationID = 0x00.
GPD IEEE address	IEEE Address	Any Valid	The identifier of the GPD entity to which the ASDU will be sent if ApplicationID = 0x02.
Green Power CommandID	Integer	0x00 – 0xff	The identifier of the command from the Green Power specification [B4], section A.4, which defines the application semantics of the ASDU.

Name	Type	Valid Range	Description
ASDULength	Integer	0x00 – (<i>aMax-MACFrameSize</i> - 9)	The number of octets in the ASDU to be transmitted.
ASDU	Set of octets	-	The set of octets forming the ASDU to be transmitted.
ASDUHandle	Integer	0x00 – 0xff	An integer handle associated with the ASDU to be transmitted.
gpTxQueueEntry-Lifetime	Unsigned 16-bit integer	0x0000 – 0xffff	The lifetime of this packet in the gpTxQueue, in milliseconds. For GPD Commissioning Reply command, initialize to Commissioning Window. 0x0000 indicates immediate transmission. 0xFFFF indicates infinity.

13294

G.2.4.2 When Generated

13295 This primitive is generated by the local application entity (GPEP) when it wishes to address a Green Power
 13296 Device Frame to the GPD identified by the GPD SrcID/GPD IEEE address parameter.

13298

G.2.4.3 Effect on Receipt

13299 On receipt of the GP-DATA.request primitive by the Inter-PAN APS, the Inter-PAN APS will construct a
 13300 Green Power Device Frame formatted as specified in section G.3.2.2, with Protocol Version sub-field and
 13301 Frame Type sub-field of the Network Frame control field set as specified in section G.3.2, containing the
 13302 given ASDU and protect it, as specified in section G.5. The stub queues the GPDF in the gpTxQueue, as
 13303 defined in section G.4.3, and later transmits the GPDF, as specified in section G.4.4, using the
 13304 MCPS-DATA.request primitive of the MAC sub-layer.
 13305

13306 **G.2.5 The INTRP-DATA.confirm Primitive**

13307 The INTRP-DATA.confirm primitive allows the Inter-PAN APS to inform the application entity about the
 13308 status of a data request.

13309 **G.2.5.1 Semantics of the Service Primitive**

13310 The primitive interface is as follows:

13311	INTRP-DATA.confirm	{
13312		ASDUHandle
13313		Status
13314		}

13315 Table G.3 defines the parameters of the INTRP-DATA.confirm primitive.

13316 **Table G.3 Parameters of the INTRP-DATA.confirm**

Name	Type	Valid Range	Description
ASDUHandle	Integer	0x00 – 0xFF	An integer handle associated with the transmitted data frame.
Status	Enumeration	Any status value returned by the MAC.	The status of the ASDU transmission corresponding to ASDUHandle returned by the MAC.

13317 **G.2.5.2 When Generated**

13318 This primitive is generated by the Inter-PAN APS on a ZigBee device and passed to the application in re-
 13319 sponse to the receipt of a MCPS-DATA.confirm primitive that is a confirmation of a previous
 13320 MCPS-DATA.request issued by the Inter-PAN APS.

13321 **G.2.5.3 Effect on Receipt**

13322 As a result of the receipt of this primitive, the application is informed of the results of an attempt to send a
 13323 frame via the Inter-PAN APS.

13324 **G.2.6 The GP-DATA.confirm Primitive**

13325 The GP-DATA.confirm primitive allows the Inter-PAN APS to inform the application entity about the status
 13326 of a Green Power data request.

G.2.6.1 Semantics of the Service Primitive

13327
13328 The primitive interface is as follows:

```
13329   GP-DATA.confirm {  
13330     ASDUHandle  
13331     Status  
13332 }
```

13333 Table G.4 defines the parameters of the GP-DATA.confirm primitive.

13334 **Table G.4 Parameters of the GP-DATA.confirm**

Name	Type	Valid Range	Description
ASDUHandle	Integer	0x00 – 0xFF	An integer handle associated with the transmitted data frame.
Status	Enumeration	Any status value returned by the MAC.	The status of the ASDU transmission corresponding to ASDUHandle as returned by the MAC. In addition to the values returned by the MAC layer, it can have the following values: TX_QUEUE_FULL ENTRY_REPLACE ENTRY_ADDED ENTRY_EXPIRED ENTRY_REMOVED FRAME_SENDING_FINALIZED

G.2.6.2 When Generated

13335
13336 This primitive is generated by the Inter-PAN APS on a ZigBee device and passed to the application (GPEP)
13337 in response to the receipt of a GP-DATA.request and MCPS-DATA.confirm primitive that is a confirmation
13338 of a previous MCPS-DATA.request issued by the Inter-PAN APS. The reasons for the various Status codes
13339 are described in section G.4.3.

G.2.6.3 Effect on Receipt

13340
13341 As a result of the receipt of this primitive, the application is informed of the results of an attempt to send a
13342 Green Power Device Frame via the Inter-PAN APS.

G.2.7 The GP-SEC.request Primitive**G.2.7.1 Semantics of the GP-SEC.request primitive**

13344
13345 The primitive interface is as follows:

```

13346    GP-SEC.request      {
13347        ApplicationID
13348        SrcID
13349        GPD IEEE Address
13350        GPDFSecurityLevel
13351        GPDFKeyType
13352        GPDSecurityFrameCounter
13353        Stub Handle
13354    }

```

13355

13356 Table G.5 defines the parameters of the GP-SEC.request primitive.

13357

13358

Table G.5 Parameters of the GP-SEC.request

Name	Type	Valid Range	Description
ApplicationID	8 bit enumeration	0x00, 0x02	ApplicationID of the Green Power Device from which the Green Power Device Frame was received. ApplicationID 0x00 indicates the usage of the 32 bit SrcID and ApplicationID 0x02 indicates the usage of the GPD IEEE address.
SrcID	Unsigned 32-bit integer	0x00000001 – 0xffffffff	The identifier of the GPD entity from which the Green Power Device Frame was received if ApplicationID = 0x00.
GPD IEEE Address	64-bit address	Any valid	The identifier of the GPD entity from which the Green Power Device Frame was received if ApplicationID = 0x02.
GPDFSecurityLevel	8-bit enumeration	0x00 – 0x03	The security level of the received frame.

Name	Type	Valid Range	Description
GPDFKeyType	8-bit enumeration	0x00 – 0x07	The security key type of the received frame.
GPD security frame counter	Unsigned 8-bit or 32-bit Integer	As specified by the GPDFSecurityLevel parameter	The security frame counter value corresponding to the received frame.
Stub Handle	Unsigned 8-bit Integer	0x00 – 0xff	The handle used between the Inter-PAN APS and the higher layers, to match the request with the response.

13359

13360 **G.2.7.2 When Generated**13361 This primitive is generated by the Inter-PAN APS and passed up on reception of a Green Power Device
13362 Frame.13363 **G.2.7.3 Effect on Receipt**13364 Upon receipt of this primitive the device is informed of reception of a Green Power Device Frame. The
13365 device then can retrieve security material for handling the frame.

13366

13367 **G.2.8 The GP-SEC.response Primitive**13368 **G.2.8.1 Semantics of the GP-SEC.response primitive**

13369 The primitive interface is as follows:

13370 GP-SEC.response	{
13371	Status
13372	Stub Handle
13373	ApplicationID
13374	SrcID
13375	GPD IEEE Address GPDFSecurityLevel

```

13376          GPDFKeyType
13377          GPDKey
13378          GPDSecurityFrameCounter
13379          SecurityWindow
13380      }

```

13381
13382 Table G.6 defines the parameters of the GP-SEC.response primitive.
13383

Table G.6 Parameters of the GP-SEC.response Primitive

Name	Type	Valid Range	Description
Status	8-bit enumeration	Any valid	The status code as returned by the end point. The following values are supported: MATCH DROP_FRAME PASS_UNPROCESSED
Stub Handle	Unsigned 8-bit Integer	0x00 – 0xff	The handle used between the Inter-PAN APS and the higher layers, to match the request with the response.
ApplicationID	8 bit enumeration	0x00, 0x02	ApplicationID of the Green Power Device from which the Green Power Device Frame was received. ApplicationID 0x00 indicates the usage of the 32 bit SrcID and ApplicationID 0x02 indicates the usage of the GPD IEEE Address.
SrcID	Unsigned 32-bit integer	0x00000001 – 0xffffffff	The identifier of the GPD entity from which the Green Power Device Frame was received if ApplicationID = 0x00.

Name	Type	Valid Range	Description
GPD IEEE Address	64-bit address	Any valid	The identifier of the GPD entity from which the Green Power Device Frame was received if ApplicationID = 0x02.
GPDFSecurityLevel	8-bit enumeration	0x00 – 0x03	The security level to be used for security processing.
GPDFKeyType	8-bit enumeration	0x00 – 0x07	The security key type to be used for security processing.
GPD Key	Security Key	Any valid	The security key to be used for GPDF security processing.
GPD security frame counter	Unsigned 32-bit Integer	Any valid	The security frame counter value to be used for security processing.
SecurityWindow	Unsigned 8-bit Integer	0x00 – 0xff	The SecurityWindow value to be used for security processing of this incoming frame.

13386 **G.2.8.2 When Generated**

13387 This primitive is generated by the Green Power endpoint (GPEP) and passed to the Inter-PAN APS on re-
13388 ception of a GP-SEC.request. The GPEP responds with appropriate status, based on the GPEP client/server
13389 functionality, the operational/commissioning mode the GPEP is in and the content of Proxy/Sink Table.

13390

13391 **G.2.8.3 Effect on Receipt**

13392 Upon receipt of this primitive the Inter-PAN APS checks the value of the *Status* field. If the *Status* is
13393 MATCH, the Inter-PAN APS triggers security processing of the GPDF, with the supplied parameters. If the
13394 *Status* is DROP_FRAME, it silently drops the frame. If the *Status* is PASS_UNPROCESSED, it generates
13395 GP-DATA.indication with the unprocessed fields GPD CommandID, GPD Command Payload and MIC
13396 copied from the received GPDF, and passes it to the Green Power application endpoint.

13397

13398 **G.2.9 The INTRP-DATA.indication Primitive**

13399 The INTRP-DATA.indication primitive allows the Inter-PAN APS to inform the next higher layer that it has
13400 received a frame that was transmitted via the Inter-PAN APS on another device.

13401 **G.2.9.1 Semantics of the Service Primitive**

13402 The primitive interface is as follows:

13403

13404	INTRP-DATA.indication	{
13405		SrcAddrMode
13406		SrcPANId
13407		SrcAddress
13408		DstAddrMode
13409		DstPANId
13410		DstAddress
13411		ProfileId
13412		ClusterId
13413		ASDULength
13414		ASDU
13415		LinkQuality
13416		}

13417

13418 Table G.7 defines the parameters of the INTRP-DATA.indication primitive.

13419

13420

Table G.7 Parameters of the INTRP-DATA.indication Primitive

Name	Type	Valid Range	Description
------	------	-------------	-------------

Name	Type	Valid Range	Description
SrcAddrMode	Integer	0x00- 0x03	The source addressing mode for the MPDU to be sent. The following values are allowed: 0x00 – no address (SrcPANId and SrcAddress omitted) 0x01 = reserved 0x02 = 16 bit short address 0x03 = 64 bit extended address
SrcPANId	16-bit PAN Id	0x0000 – 0xffff	The 16-bit PAN identifier of the entity from which the ASDU is being transferred.
SrcAddress	64-bit address	As specified by the SrcAddrMode parameter	The device address of the entity from which the ASDU is being transferred.
DstAddrMode	Integer	0x00 – 0x03	The addressing mode for the destination address used in this primitive. This parameter can take one of the values from the following list: 0x00 = no address (DstPANId and DstAddr omitted) 0x01 = 16-bit group address 0x02 = 16-bit NWK address, usually the broadcast address 0xffff 0x03 = 64-bit extended address
DstPANID	16-bit PAN Id	0x0000 – 0xffff	The 16-bit PAN identifier of the entity or entities to which the ASDU is being transferred or the broadcast PAN ID 0xffff.
DstAddress	16-bit or 64-bit address	As specified by the DstAddrMode parameter	The address of the entity or entities to which the ASDU is being transferred.

Name	Type	Valid Range	Description
ProfileId	Integer	0x0000 – 0xffff	The identifier of the application profile for which this frame is intended.
ClusterId	Integer	0x0000 – 0xffff	The identifier of the cluster, within the profile specified by the ProfileId parameter, which defines the application semantics of the ASDU.
ASDULength	Integer	0x00 – (<i>aMax-MACFrameSize</i> - 9)	The number of octets in the ASDU to be transmitted.
ASDU	Set of octets	-	The set of octets forming the ASDU to be transmitted.
LinkQuality	Integer	0x00 – 0xff	The link quality observed during the reception of the ASDU.

13421

G.2.9.2 When Generated13422
13423
13424

This primitive is generated and passed to the application in the event of the receipt, by the Inter-PAN APS, of a MCPS-DATA.indication primitive from the MAC sub-layer, containing a frame that was generated by the Inter-PAN APS of a peer ZigBee device, and that was intended for the receiving device.

13425 **G.2.9.3 Effect on Receipt**

13426 Upon receipt of this primitive the application is informed of the receipt of an application frame transmitted,
13427 via the Inter-PAN APS, by a peer device and intended for the receiving device. The values of the IN-
13428 TRP-DATA.indication shall be copied into the matching field names of the APSME-DATA.indication.
13429 Additionally these fields shall be set as follows:

- 13430 1. The DstAddrMode shall be set to 0x04.
- 13431 2. The DstAddress shall be set to the DstAddress of the INTRP-DATA.indication primitive.
- 13432 3. The SrcAddrMode shall be set to 0x04.
- 13433 4. The SrcAddress shall be set to the SrcAddress of the INTRP-DATA.indication primitive.
- 13434 5. The SecurityStatus field enumeration shall be set to UNSECURED.
- 13435 6. The Inter-PAN field shall be set to TRUE.

G.2.10 The GP-DATA.indication Primitive

13437 The GP-DATA.indication primitive allows the Inter-PAN APS to inform the next higher layer that it has
13438 received a Green Power Device Frame.

G.2.10.1 Semantics of the Service Primitive

13440 The primitive interface is as follows:

13442	GP-DATA.indication	{
13443		Status
13444		LinkQuality
13445		SeqNumber
13446		SrcAddrMode
13447		SrcPANId
13448		SrcAddress
13449		ApplicationID
13450		GPDFSecurityLevel
13451		GPDFKeyType
13452		AutoCommissioning
13453		RxAfterTx
13454		SrcID
13455		GPD Security Frame Counter
13456		CommandID
13457		ASDULength
13458		ASDU
13459		MIC
13460		}

13461
13462 Table G.8 defines the parameters of the GP-DATA.indication primitive.
13463

Table G.8 Parameters of the GP-DATA.indication Primitive

Name	Type	Valid Range	Description
Status	8-bit enumeration	Any Valid	Status Code returned by Green Power. It can have the following values: SECURITY_SUCCESS NO_SECURITY COUNTER_FAILURE AUTH_FAILURE UNPROCESSED
LinkQuality	Integer	0x00 – 0xff	The link quality observed during the reception of the ASDU.
SeqNumber	Unsigned 8-bit integer	0x00 – 0xff	The sequence number from the MAC header of the received frame.
SrcAddrMode	Integer	0x00- 0x03	The source addressing mode for the MPDU to be sent. The following values are allowed: 0x00 – no address (SrcPANId and SrcAddress omitted) 0x01 = reserved 0x02 = 16 bit short address 0x03 = 64 bit extended address
SrcPANId	16-bit PAN Id	0x0000 – 0xffff	The 16-bit PAN identifier of the entity from which the ASDU is being transferred.
SrcAddress	64-bit address	As specified by the SrcAddrMode parameter	The device address of the entity from which the ASDU is being transferred.

Name	Type	Valid Range	Description
ApplicationID	8-bit enumeration	0x00, 0x02	The ApplicationID, corresponding to the received frame. ApplicationID 0x00 indicates the use of a SrcID; ApplicationID 0x02 indicates the usage of the GPD IEEE address.
GPDFSecurityLevel	8-bit enumeration	0x00- 0x03	The security level of the received frame.
GPDFKeyType	8-bit enumeration	0x00 – 0x07	The security key type, which was successfully used for security processing the received frame.
Auto-Commissioning	Boolean	TRUE/FALSE	The Auto-commissioning sub-field, copied from the received frame.
RxAfterTx	Boolean	TRUE/FALSE	The RxAfterTx sub-field, copied from the received frame.
SrcID	Unsigned 32-bit Integer	0x00000000 – 0xffffffff	The identifier of the GPD entity from which the frame was received if ApplicationID = 0x00.

Name	Type	Valid Range	Description
GPD security frame counter	Unsigned 32-bit Integer	As specified by the GPDFSecurityLevel parameter	The security frame counter value used on transmission by the GPD entity from which the frame was received.
GPD Command ID	Unsigned 8-bit Integer	0x00 – 0xff	The identifier of the command, within the Green Power specification [B4] section A.4 which defines the application semantics of the frame.
ASDULength	Integer	0x00 – (<i>aMaxPHYPacketSize</i>)	The number of octets in the received ASDU.
ASDU	Set of octets	-	The set of octets in the received ASDU.
MIC	Unsigned 16-bit or 32-bit Integer	As specified by the GPDFSecurityLevel parameter	The set of octets forming the MIC for the received frame.

13465

G.2.10.2 When Generated13466
13467

This primitive is generated and passed to the application in the event of the receipt, by the Inter-PAN APS, of a MCPS-DATA.indication primitive from the MAC sub-layer, containing a Green Power Device Frame.

13468

G.2.10.3 Effect on Receipt13469
13470

Upon receipt of this primitive the Green Power endpoint (GPEP) is informed of the receipt of a Green Power Device Frame.

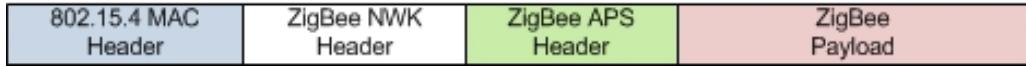
G.2.11 Qualifying and Testing of Inter-PAN Messages

Support for Inter-PAN messages and Green Power is optional. If a device claims Inter-PAN communication support then certification and application level testing shall ensure both the sending and receiving devices correctly react and understand the INTRP-DATA.request and INTRP-DATA.indication primitives. Green Power certification and application level testing shall also ensure the GP-DATA.request, GP-DATA.indication, GP-SEC.request, and GP-SEC.response primitives are supported as mandated by the Green Power Specification [B4].

G.3 Frame Formats

The overall view of a ZigBee frame is as shown in Figure G.2

Figure G.2 - ZigBee Frame Format Overview



Briefly, the frame contains the familiar headers controlling the operation of the MAC sub-layer, the NWK layer and the APS. Following these, there is a payload, formatted as specified in [B1].

Since most of the information contained in the NWK header is not relevant for Inter-PAN transmission, the Inter-PAN frame, shown in Figure G.3, contains only a stub of the NWK header. A Inter-PAN APS header is also used and is described in section G.3.3.

Figure G.3 Inter-PAN Frame Format

Octets: 2	1	variable	2
Frame Control	Sequence Number	Addressing Fields	NWK Frame Control
802.15.4 MAC Header			NWK Header

For Green Power Device Frames there is a different set of MAC and NWK headers as shown in Figure G.4.

Figure G.4 Green Power Device Frame Format

Octets: 2	1	4/10/12/ Variable	1	0/1	0/4	0/4	Variable	0/2/4
Frame Control	Sequence Number	Addressing Fields	NWK Frame Control	Extended NWK Frame Control	GPD SrcID	Security Frame Counter	GPDF Application Payload	MIC
802.15.4 MAC Header			GPDF NWK Header				GPDF Application Payload	GPDF NWK Trailer

13493 **G.3.1 MAC Header**

13494 The 802.15.4 MAC header has several options depending on how the frame is being used. The MAC header
 13495 fields are shown in Table G.9 with notes on their use.

13496 **Table G.9 MAC Header Fields for Inter-PAN APS Frames**

Field Name	Octets	Usage
Frame Control	2	Varies by Inter-PAN APS frame
Sequence Number	1	Normally used as MAC sequence number, increasing for each frame sent. Green Power usage discussed in section G.3.2.2.1.
Destination PAN ID	0/2	May be set as the PANID of the destination or 0xffff.
Destination Address	2/8	Normally either broadcast short address or a 64 bit long address of the destination. Green Power usage discussed in section G.3.1.2
Source PAN ID	0/2	Used in Inter-PAN messaging but not in Green Power Device Frames
Source Address	2/8	Normally set to the 64 bit address of the source device. Green Power usage discussed in section G.3.1.2

13498 The MAC header usage varies by application using the Inter-PAN messaging.

13499 **G.3.1.1 MAC Header usage for Inter-PAN messaging**

13500 Because Inter-PAN messaging is used for devices not on the ZigBee network, short addressing is not normally
 13501 used unless it is the broadcast short address such that any device within range can respond. Otherwise
 13502 the 64 bit long addresses are used for source and destination addressing. Source and Destination PANID's
 13503 may be used or may be omitted.

13504 **G.3.1.2 MAC Header usage for Green Power Device Frames**

13505 The Green Power Device Frame originating from the GPD can be sent with MAC Dest PANID and MAC
 13506 Dest Address set to 0xffff.

13507 If the IEEE address of the GPD is used for unique identification, the Green Power Device Frame shall include
 13508 the Extended NWK Frame Control field and its ApplicationID sub-field shall be set to 0b010. Then, for the
 13509 frame transmitted by the GPD, the GPD's IEEE address shall be transmitted in the MAC Src Address field,
 13510 and the Intra-PAN sub-field and the Source Addressing Mode sub-field of the MAC Frame Control field shall
 13511 be set accordingly. For the frames transmitted to the GPD, the GPD's IEEE address shall be transmitted in the
 13512 MAC Dest Address field, and the Intra-PAN sub-field and the Destination Addressing Mode sub-field of the
 13513 MAC Frame Control field shall be set accordingly; see also section G.3.2.

13514

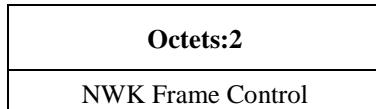
13515 **G.3.2 Network Header**

13516 **G.3.2.1 Stub NWK Header for Inter-PAN Messages**

13517 The stub NWK Header for Inter-PAN messages is shown below in Figure G.5.

13518

13519 **Figure G.5 Stub NWK Header for Inter-PAN messages**



13520 The NWK header Frame control field for the Inter-PAN messages is formatted exactly as the NWK header
13521 used by other ZigBee frames, see section 3.3.1.1 of the current specification.

13522 For Inter-PAN messages, the frame type 0b11 is used with the protocol version of the ZigBee stack. All
13523 other sub-fields shall have a value of 0.

13524 **G.3.2.2 Stub NWK Header for Green Power Device Frames**

13525 **G.3.2.2.1 Stub NWK for Green Power Device Frames**

13526 The format of the stub NWK Header for GPDF is formatted as shown in Figure G.6.

13527

Figure G.6 NWK Header Frame Control for Green Power Device Frames

Octets: 1	0/1	0/4	0/4	Variable	0/2/4
NWK frame control	Extended NWK Frame Control	GPD SrcID	Security frame counter	GPDF application payload	MIC
GPDF NWK Header				GPDF application payload	GPDF NWK Trailer

13528 The NWK Frame Control of the stub NWK Header for GPDF is formatted as shown in **Error! Reference
13529 source not found..**

13530

13531

13532

Figure G.7 NWK Header Frame Control for Green Power Device Frames

NWK Frame Control				Extended NWK Frame Control	
Bits: 0-1	2-5	6	7	Bits 0-2	3-7
Frame type	Protocol version	Auto-Commissioning	NWK Frame Control Extension	Application ID	Defined for specific ApplicationID

13533

13534 The sub-fields of the NWK frame control field are as follows:

13535 For the Green Power Device Frames, the ZigBee Protocol Version sub-field shall carry the value of 0x3.
 13536 The frame type sub-field as used in combination with the ZigBee Protocol Version = 0x3, can take the values
 13537 specified in Table G.10.

Table G.10 Values for Frame Type for GPDF

Value	Description
0b00	Data Frame
0b01	Maintenance Frame
0b10	Reserved
0b11	Reserved

13539
 13540 If the Frame Type 0b01 (Maintenance frame) is used then the Green Power SrcID and the security fields
 13541 (security frame counter and MIC) shall not be present. Green Power Devices should omit the extended
 13542 NWK frame control and it may also be omitted when sending to Green Power Devices. The NWK Frame
 13543 Control extension sub-filed shall be set accordingly.

13544 If the Frame Type 0b00 (Data frame) is used the Green Power frame format is as follows:

13545 The Auto Commissioning sub-field indicates if the device implements the GPD Commissioning command
 13546 (see reference [B4], section A.4). If set to 0b1 the device does not implement the GPD Commissioning
 13547 command. If set to 0b0 the device does implement the GPD Commissioning command.

13548 The NWK Frame Control Extension, if set to 0b1, indicates the Extended NWK Frame Control field is
 13549 present. The Extended NWK Frame Control extension shall be present if the ApplicationID is different than
 13550 0b000.

13551 The ApplicationID allows for re-defining the frame format. The current specification defines the frame
 13552 format for ApplicationID 0b000 and 0b010 (Green Power). Default value to be used on reception, if the
 13553 Extended NWK Frame Control field is not present is 0b000.

13554 For ApplicationID 0b000 and 0b010 and 0b001, the bits 3-7 are defined in Figure G.8. For ApplicationID
 13555 0b000 the Extended NWK Frame Control field shall be present if the frame is protected, if RxAfterTx is set,
 13556 or if the frame is sent to the Green Power Device.

13557

13558 **Figure G.8 Format of Extended NWK Frame Control field for GPDF with Application ID 0b000 and 0b010**

Bits 3-4	5	6	7
Security Level	Security Key	RxAfterTx	Direction

13559
 13560 The SecurityLevel sub-field indicates if the frame is protected.
 13561 If ApplicationID is set to 0b000 and 0b010, the Security Level sub-field can have values as defined in Table
 13562 G.10. Default value to be used on reception, if the Extended NWK Frame Control field is not present, is
 13563 0b00.
 13564 If the SecurityLevel is set to 0b00, the SecurityKey sub-field is ignored on reception, and the fields Security
 13565 frame counter and MIC are not present. The MAC sequence number field carries the random or the incre-
 13566 mental sequence number, according to the capabilities of this GPD.

13567 If the SecurityLevel is set to 0b01, the Security Frame counter field is not present, the MAC sequence number
 13568 field carries the 1LSB of the frame counter, and the MIC field is present, has the length of 2B, and carries the
 13569 2LSB of the Message Integrity Code (see section G.5.4 of the current document).

13570 If the SecurityLevel is set to 0b10 or 0b11, the Security Frame counter field is present, has the length of 4B,
 13571 and carries the full 4B security frame counter, the MIC field is present, has the length of 4B, and carries the
 13572 full 4B Message Integrity Code (see section G.5.4 of the current document). The MAC sequence number
 13573 field carries the random or the incremental sequence number, according to the capabilities of this GPD; it
 13574 shall not be used for security, but only for duplicate filtering at MAC level.

13575

Table G.11 Values of gpSecurityLevel

Value	Description
0b00	No security
0b01	1LSB of frame counter and short (2B) MIC only
0b10	Full (4B) frame counter and full (4B) MIC only
0b11	Encryption & full (4B) frame counter and full (4B) MIC

13577

13578 The SecurityKey sub-field indicates the type of the key used for frame protection by this GPD. The Security
 13579 Key sub-field, if set to 0b1, indicates an individual key (KeyType 0b100 or 0b111). If set to 0b0, it indicates
 13580 a shared key (KeyType 0b011, 0b010 or 0b001) or no key.

13581 The RxAfterTx sub-field is a Boolean flag. If the value of this sub-field is 0b1, then it indicates that the GPD
 13582 will enter the receive mode after gpdRxOffset, for a device-specific duration, but not shorter than
 13583 gpdMinRxWindow. If the value of this sub-field is 0b0, then the GPD will not enter the receive mode after
 13584 sending this particular GPDF frame. Default value to be used on reception, if the Extended NWK Frame
 13585 Control field is not present, is 0b0.

13586 The Direction sub-field shall be set to 0b0, if the GPDF is transmitted by the GPD, and to 0b1, if the GPDF is
 13587 transmitted by GPP. Default value to be used on reception, if the Extended NWK Frame Control field is not
 13588 present, is 0b0.

G.3.2.2.2 Remaining Fields of the Stub NWK Header for GPDF

13590 The GPDSrcID field is present if the FrameType sub-field is set to 0b00 and the ApplicationID sub-field of
 13591 the Extended NWK Frame Control field is set to 0b000 (or not present). It is also present if the FrameType
 13592 sub-field is set to 0b01, the NWK Frame control Extension sub-field is set to 0b1, and the ApplicationID
 13593 sub-field of the Extended NWK Frame Control field is set to 0b000. The GPDSrcID field carries the unique
 13594 identifier of the GPD, to/by which this GPDF is sent. The value of 0x00000000 indicates unspecified. The
 13595 value of 0xffffffff indicates all. The values 0xffffffff9 – 0xffffffffe are reserved. The GPDSrcID field is not
 13596 present if the FrameType sub-field is set to 0b01 and the Extended NWK Frame control sub-field is set to
 13597 0b0. Unique identification of the GPD by an address is not required then. The GPDSrcID field is not present
 13598 if the ApplicationID sub-field of the Extended NWK Frame Control field is set to 0b010. The GPD is then
 13599 identified by its IEEE address, which is then carried in the corresponding MAC address field, source or
 13600 destination for the GPDF sent by or to the GPD, respectively.

13601 The presence and length of the Security frame counter field is dependent on the value of ApplicationID and
 13602 SecurityLevels sub-field, as described above.

13603 The MIC field carries the Message Integrity Code for this message, calculated as specified in section G.5.4 of
 13604 the current specification. Its presence and length is dependent on the value of ApplicationID and Secu-
 13605 rityLevel sub-fields, as described above.

13606 The application payload of the GPDF is defined in [B4], section A.1.4.1.6.

G.3.3 Inter-PAN APS Header

13608 The format of the Inter-PAN APS header is shown in Figure G.9. This is used in normal Inter-PAN messages
13609 and Touchlink messages but not in Green Power Device Frames.

13610

13611 **Figure G.9 Inter-PAN APS Header Format**

Octets: 1	0/2	2	2
APS frame control	Group address	Cluster identifier	Profile identifier
Addressing fields			

13612

13613 The Inter-PAN APS header contains only 4 fields totaling a maximum of 7 octets in length.

13614 The APS frame control field shall be 1 octet in length and is identical in format to the frame control field of
13615 the general APDU frame in [B3] (see Figure G.10).

13616

13617 **Figure G.10 Format of the APS Frame Control Field for Inter-PAN Messages**

Bits: 0-1	2-3	4	5	6	7
Frame type	Delivery Mode	Reserved	Security	ACK request	Extended Header Present

13618

13619 The fields of the frame control field have the following values:

13620

- The frame type sub-field shall have a value of 0b11, which is the Inter-PAN APS frame type.
- The delivery mode sub-field may have a value of 0b00, indicating unicast, 0b10, indicating broadcast or 0b11 indicating group addressing.
- Security is never enabled for Inter-PAN transmissions. This sub-field shall be a value of 0.
- The ACK request sub-field shall have a value of 0, indicating no ACK request. No APS ACKs are to be used with Inter-PAN transmissions.
- The extended header present sub-field shall always have a value of 0, indicating no extended header.

13627
13628

The optional group address shall be present if and only if the delivery mode field has a value of 0x0b11. If present it shall contain the 16-bit identifier of the group to which the frame is addressed.

13629
13630
13631

The cluster identifier field is 2 octets in length and specifies the identifier of the cluster to which the frame relates and which shall be made available for filtering and interpretation of messages at each device that takes delivery of the frame. For touchlink this has a value of 0x1000.

13632
13633
13634

The profile identifier is two octets in length and specifies the ZigBee profile identifier for which the frame is intended and shall be used during the filtering of messages at each device that takes delivery of the frame. For touchlink this has the value of 0xc05e.

13635

G.4 Frame Processing

Assuming the INTRP-SAP described above, frames transmitted using the Inter-PAN APS are processed as described here.

G.4.1 Inter-PAN Transmission (non Green Power Device Frames)

On receipt of the INTRP-DATA.request primitive, the Inter-PAN APS shall construct a Inter-PAN APS frame. The header of the Inter-PAN APS frame shall contain a NWK and an APS frame control field as described in section G.3, a cluster identifier field equal to the value of the ClusterId parameter of the INTRP-DATA.request and a profile identifier field equal to the value of the ProfileId parameter. If the DstAddrMode parameter of the INTRP-DATA.request has a value of 0x01, indicating group addressing, then the APS header shall also contain a group address field with a value corresponding to the value of the DstAddress parameter. The payload of the Inter-PAN APS frame shall contain the data payload to be transmitted.

The Inter-PAN APS frame will then be transmitted using the MCPS-DATA.request primitive of the MAC sub-layer with key primitive parameters set as follows:

- The value of the SrcAddrMode parameter of the MCPS-DATA.request shall always be set to a value of three, indicating the use of the 64-bit extended address.
- The SrcPANId parameter shall be equal to the value of the macPANID attribute of the MAC PIB.
- The SrcAddr parameter shall always be equal to the value of the MAC sub- layer constant aExtendedAddress.
- If the DstAddrMode parameter of the INTRP-DATA.request primitive has a value of 0x01, then the DstAddrMode parameter of the MCPS-DATA.request shall have a value of 0x02. Otherwise, the DstAddrMode parameter of the MCPS-DATA.request shall reflect the value of the DstAddrMode parameter of the INTRP-DATA.request.
- The DstPANId parameter shall have the value given by the DstPANID parameter of the INTRP-DATA.request primitive.
- If the DstAddrMode parameter of the INTRP-DATA.request has a value of 0x01, indicating group addressing, then the value of the DstAddr parameter of the MCPS-DATA.request shall be the broadcast address 0xffff. Otherwise, value of the DstAddr parameter shall reflect the value of the DstAddress parameter of the INTRP-DATA.request primitive.
- The MsduLength parameter shall be the length, in octets, of the Inter-PAN APS frame.
- The Msdu parameter shall be the Inter-PAN APS frame itself.
- If the transmission is a unicast, then the value of the TxOptions parameter shall be 0x01, indicating a request for acknowledgement. Otherwise, the TxOptions parameter shall have a value of 0x00, indicating no options.

On receipt of the MCPS-DATA.confirm primitive from the MAC sub-layer, the Inter-PAN APS will invoke the INTRP-DATA.confirm primitive with a status reflecting the status returned by the MAC.

13674 **G.4.2 Inter-PAN Reception (non Green Power Device**

13675 **Frames)**

13676 On receipt of the MCPS-DATA.indication primitive from the MAC sub-layer, the receiving entity - in case of
13677 a ZigBee device this is normally the NWK layer - shall determine whether the frame should be passed to the
13678 Inter-PAN APS or processed as specified in [B3]. For a frame that is to be processed by the Inter-PAN APS,
13679 the non- varying sub-fields of the NWK frame control field must be set exactly as described in section
13680 G.3.2.1and the APS frame control field must be set exactly as described in section G.3.3. Any variation
13681 from this format shall trigger the message to be dropped and no further processing shall be done.

13682 If the delivery mode sub-field of the APS frame control field of the Inter-PAN APS header has a value of
13683 0b11, indicating group addressing, then, if the device implements group addressing, the value of the group
13684 address field shall be checked against the NWK layer group table, and, if the received value is not
13685 present in the table, the frame shall be discarded with no further processing or action.

13686 On receipt of a frame for processing, the Inter-PAN APS shall generate an INTRP- DATA.indication with
13687 parameter values as follows:

- 13688 • The value of the SrcAddrMode parameter of the INTRP-DATA.indication shall always be set to a
13689 value of three, indicating the use of the 64-bit extended address
- 13690 • The value of the SrcPANId parameter shall reflect that of the SrcPANId parameter of the
13691 MCPS-DATA.indication.
- 13692 • The SrcAddress parameter of the INTRP-DATA.indication shall always reflect the value of a 64-bit
13693 extended address.
- 13694 • Values for the DstAddrMode parameter shall be one of:
 - 13695 ◦ 0x03, if the DstAddrMode parameter of the INTRP-DATA.indication has a value of 0x03.
 - 13696 ◦ 0x02, if the DstAddrMode parameter of the INTRP-DATA.indication has a value of 0x02
- 13697 • The value of the DstPANId parameter of the INTRP-DATA.indication shall reflect the value of the
13698 DstPANId parameter of the MCPS-DATA.indication.
- 13699 • If the DstAddrMode parameter of the INTRP-DATA.indication has a value of 0x01, indicating
13700 group addressing then the DstAddress parameter of the INTRP-DATA.indication shall reflect the
13701 value of the group address field of the Inter-PAN APS header. Otherwise, the value of the
13702 DstAddress parameter of the INTRP-DATA.indication shall reflect the value of the DstAddr pa-
13703 rameter of the MCPS-DATA.indication.
- 13704 • The value of the ProfileId parameter shall be the same as the value of the profile identifier field of
13705 the Inter-PAN APS header.
- 13706 • The value of the ClusterId parameter shall be the same as the value of the cluster identifier field of
13707 the Inter-PAN APS header.
- 13708 • The ASDULength field shall contain the number of octets in the Inter-PAN APS frame payload.
- 13709 • The ASDU shall be the Inter-PAN APS payload itself.
- 13710 • The value of the LinkQuality parameter shall reflect the value of the mpduLinkQuality parameter of
13711 the MCPS-DATA.indication.

13712 **G.4.3 Green Power Device Frame Transmission**

13713 On receipt of the GP-DATA.request primitive, the Inter-PAN APS shall check the gpTxQueue. If the
13714 gpTxQueue already has an entry for the GPD ID (i.e. GPD SrcID/GPD IEEE address) in the
13715 GP-DATA.request, the previous GPDF is overwritten and GP-DATA.confirm with the Status EN-
13716 TRY_REPLACE is provided to the GPEP. If the gpTxQueue has no previous entry for this GPD
13717 SrcID/GPD IEEE address and it has empty entries, the GPDF is added to the gpTxQueue and
13718 GP-DATA.confirm with the Status ENTRY_ADDED is provided to the GPEP. If the gpTxQueue has no
13719 previous entry for this GPD SrcID/GPD IEEE address and it is full, the Inter-PAN APS returns
13720 GP-DATA.confirm with the Status set to QUEUE_FULL.

13721 **G.4.3.1 gpTxQueue**

13722 In gpTxQueue, GPDF are stored for transmission to GPD.

13723 In its gpTxQueue, each GP infrastructure device shall have a maximum of only one pending GPDF frame per
13724 GPD ID. Each entry in the gpTxQueue shall have a gpTxQueueEntryLifetime parameter associated, initiated
13725 by the value in the GP-DATA.request. When this timeout elapses, the GP-DATA.confirm with the Status
13726 ENTRY_EXPIRED is returned to the GPEP, the entry is cleared and can be used for any GPDF for
13727 any GPD ID. The gpTxQueue shall have a minimum length of 5 entries.

13728 **G.4.3.2 gpTxOffset**

13729 The gpTxOffset is the time after which the Inter-PAN APS shall send a GPDF in response to a GPDF with
13730 RxAfterTx sub-field set, if any present in the gpTxQueue for this GPD ID. It is measured from the start of the
13731 reception of the first GPDF in a given GPFS.

13732 The gpTxOffset has value identical to the gpdRxOffset (see sec. A.1.6.3.1).

13733 **G.4.3.3 gpTxDuration**

13734 The gpTxDuration is the maximum allowed transmission time for the Inter-PAN APS after gpTxOffset.
13735 Thus, depending on the GPDF length, the Inter-PAN APS may send the GPDF more than once, to increase
13736 the reliability of communication. It is measured from the start of the transmission of the first GPDF in a given
13737 GPFS.

13738 The gpTxDuration has the value of 10ms.

13739 **G.4.4 Green Power Device Frame Reception**

13740 On receipt of a GPDF, the Inter-PAN APS shall filter out (silently drop) frames with ApplicationID value
13741 other than 0b000 and 0b010 frames with Direction sub-field of the Extended NWK Frame Control field set to
13742 0b1, and duplicate frames. For this purpose, the MCPS-DATA.indication shall also include the MAC se-
13743 quence number parameter.

13744 Frames with ApplicationID 0b000 and 0b010 shall be further processed, as follows.

13745 The Inter-PAN APS shall check the *SecurityLevel*. If the *SecurityLevel* is not supported, the stub shall silently
13746 drop the frame. If *SecurityLevel* is supported and has the value of 0b00-0b10, and GPD CommandID has the
13747 value from the range 0xf0-0xff, the GPDF is silently dropped. If *SecurityLevel* is supported, the stub then
13748 generates GP-SEC.request and waits for GP-SEC.response.

13749 On receipt of GP-SEC.response with *Status* DROP_FRAME, the stub drops the frame. On receipt of
13750 GP-SEC.response with *Status* PASS_UNPROCESSED, the stub generates GP- DATA.indication for the
13751 unprocessed frame. On receipt of GP-SEC.response with *Status* MATCH, the stub security-processes the
13752 received GPDF, as described in section G.5.

13753 If security processing fails, the stub indicates that with GP-DATA.indication carrying the corresponding
 13754 *Status* value and stops any further processing of this frame.

13755 If security processing is successful, and the *SecurityLevel* was 0b11, the stub checks the plaintext value of the
 13756 GPD *CommandID*. If it has the value from the range 0xf0-0xff, the GPDF is silently dropped.

13757 If security processing was successful, and the GPD *CommandID* is not from the 0xf0 – 0xff range, the stub
 13758 checks if the *RxAfterTx* sub-field of the *Extended NWK Frame Control* field of the received GPDF was set to
 13759 0b1. If yes, it searches the *gpTxQueue* for an entry for this GPD ID. If a suitable GPDF is found, the stub
 13760 triggers security processing of the to-be-sent GPDF with the same security input parameters as for the re-
 13761 ceived GPDF. If the Data Frame Type is used, the NWK Frame Control Extension sub-field shall be set to
 13762 0b1, the Extended NWK Frame Control field shall be present, and the *RxAfterTx* sub-field shall be set to 0b0
 13763 and the Direction sub-field shall be set to 0b1.

13764 Then, the Inter-PAN APS constructs the GPDF with the ApplicationID sub-field of the Extended NWK
 13765 Frame Control field set to 0b000 or 0b010, as supplied in the GP-DATA.request primitive, and the remaining
 13766 fields as supplied by the GP-DATA.request primitive.

13767 The Inter-PAN APS schedules GPDF transmission to commence after *gpTxOffset*, by sending
 13768 MCPS-DATA.request, with *UseCSMA* parameter set to FALSE and *Use MAC ACK* copied from the
 13769 *TxOptions* parameter as supplied by the GP-DATA.request primitive.

13770 The parameter *UseCSMA* of the *TxOptions* is an extension to the MCPS-DATA.request and shall be
 13771 propagated by the stub to the MAC layer. When *UseCSMA* is FALSE, CSMA/CA shall be skipped for the
 13772 transmission of this GPDF. On reception of the MCPS-DATA.confirm, the stub calls GP-DATA.confirm
 13773 with *Status* value copied from the MCPS-DATA.confirm.

13774 Subsequently, and if no matching entry is found in the *gpTxQueue*, the stub indicates reception of the GPDF
 13775 to the next higher layer, by calling GP-DATA.indication. If *SecurityLevel* was 0b00, the stub calls
 13776 GP-DATA.indication with the *Status NO_SECURITY*; if *SecurityLevel* was 0b01 – 0b11, the stub calls
 13777 GP-DATA.indication with the *Status SECURITY_SUCCESS*.

13778 **G.5 Green Power Security Stub Operations**

13779 **G.5.1 Per GPDF Security Level and Key Selection**

13780 The Inter-PAN APS shall:

- 13781 • For the incoming secured GPDF: use the parameters supplied by the GP-SEC.response.
- 13782 • For the outgoing secured GPDF: use the same key and protection level as for the triggering GPDF.

13783 **G.5.2 Construction of AES Nonce**

13784 The AES nonce, defined by the current specification (see section 4.5.2.2) to have the format as depicted in
 13785 Figure G.11, is used for security operations and shall be constructed in the following way.

13786
 13787 **Figure G.11 Format of the AES Nonce for Green Power Device Frames**

Octets: 8	4	1
Source Address	Frame Counter	Security Control

13788 For *ApplicationID* = 0b000, the *Source address* parameter shall take the value:

- 13789 • for the incoming secured GPDF (i.e. the GPDF sent by the GPD): *SourceAddress[63:32]* = *SrcID*,
 13790 *SourceAddress[31:0]* = *SrcID*;

- 13791 • for the outgoing secured GPDF (i.e. the GPDF sent to the GPD): SourceAddress[63:32] = SrcID,
 13792 SourceAddress[31:0] = 0;

13793 The SrcID is little Endian (LSB first).

13794 For example, if the SrcID = 0x87654321, the Source address parameter takes the following values:

- 13795 • for the incoming secured GPDF: 0x8765432187654321 = { 0x21, 0x043, 0x65, 0x87, 0x21, 0x43,
 13796 0x65, 0x87 };
- 13797 • for the outgoing secured GPDF: 0x8765432100000000 = { 0x00, 0x00, 0x00, 0x00, 0x21, 0x43,
 13798 0x65, 0x87 }.

13799 For *ApplicationID* = 0b010, the *Source address* parameter shall take the value of the IEEE address of the
 13800 GPD, for both incoming and outgoing secured GPDF.

13801 *Frame counter* parameter shall take the value:

- 13802 • for the incoming secured GPDF: 4B frame counter for this GPD, part or whole of which is being
 13803 transmitted in the GPDF:
 - 13804 ◦ if SecurityLevel was 0b01: the frame counter value is derived as described in A.3.7.2.4
- 13805 • For the outgoing secured GPDF: the 4B value of frame counter that was last used by this GPD (i.e.
 13806 the frame counter value from the GPDF received from this GPD with RxAfterTx=TRUE that im-
 13807 mediately precedes the sending of this frame to the GPD).

13808 *Security control* field, defined to be part of the AES nonce by the current specification and formatted as
 13809 shown in Figure G.12, is never exchanged between the GP-capable devices. Thus, for interoperability, the
 13810 values used shall be as defined below.

13811 **Figure G.12 Format of the Security Control field of the AES Nonce for Green Power Device Frames**

Bits: 0-2	3-4	5	6-7
Security level	Key Identifier	ExtendedNonce	Reserved

- 13813 • Security level = 0b101
- 13814 • Key identifier = 0b00
- 13815 • Note that this security level and Key identifier are never transmitted and are NOT used for deter-
 13816 mining the transformation applied to the packet, since those are governed by the Security sub- field
 13817 of the NWK Frame Control field of the GPDF. The values here are defined for interoperability only.
- 13818 • Extended nonce = 0b0
- 13819 • Reserved =
 - 13820 ◦ For *ApplicationID* = 0b000 and for incoming secured GPDF (i.e. GPDF sent by GPD):
 13821 Reserved = 0b00;
 - 13822 ◦ For outgoing secured GPDF (i.e. GPDF sent to GPD) with an *ApplicationID* = 0b010:
 13823 Reserved = 0b11.

13825 The *Nonce* shall be formatted little endian, i.e. LSB first. Also the fields *Source address* and *Frame counter*
 13826 shall be little endian, i.e. LSB first.

13828 **G.5.3 Initialization**

13829 If the *SecurityLevel* field of the GPDF has the value 0b01, the following transformation applies.

13830 The definition Payload is applied to the following fields of the GPDF:

13831 Payload = GPD CommandID || GPD Command Payload.

13832 The definition Header is applied to the following fields of the GPDF:

13833 Header = MAC sequence number || MAC addressing fields || NWK Frame Control || Extended NWK
13834 Frame Control || SrcID.

13835 The following definitions apply:

- 13836 • For the MAC sequence number field as part of the Header
 - 13837 ◦ In case of an incoming frame, the MAC sequence number from the received frame is used.
 - 13838 ◦ In case of an outgoing frame, 1LSB of the Security Frame Counter is used for security
13839 processing. Note: the 1LSB of the Security Frame Counter is independent of the macDSN
13840 attribute the MAC layer will use to transmit the frame.
- 13841 • MAC addressing fields = are as in the received frame / as requested by the application;
- 13842 • SrcID field = as in the received frame / as requested by the application (i.e. only for ApplicationID =
13843 0b000).

13844 If the *SecurityLevel* field of the GPDF has the value 0b10 or 0b11, the following transformation applies.

13845 The definition Payload is applied to the following fields of the GPDF:

13846 Payload = GPD CommandID || GPD Command Payload.

13847 The definition Header is applied to the following fields of the GPDF:

13848 Header = NWK Frame Control || Ext NWK Frame Control || SrcID || Frame counter; whereby the
13849 SrcID field is only present if the ApplicationID = 0b000.

13850 **G.5.4 Outgoing frame encryption and authentication**

13851 Determine the security level, as described in section G.5.1, and perform initialization, as described in section
13852 G.5.3.

13853 **G.5.4.1 CCM* execution**

13854
13855 Execute the CCM* mode encryption and authentication operation, as specified in Annex A. The following
13856 parameters are used:

- 13857 • The parameter M is =4, which means that 4B MIC is calculated (irrespective of *gpdSecurityLevel*).
- 13858 • Nonce is constructed as described in section G.5.2.
- 13859 • The bit string Key determined as described in section G.4.4.
- 13860 • If the frame requires encryption (as indicated by *gpdSecurityLevel* = 0b11),
 - 13861 ◦ the octet string a shall be the Header, as defined in section G.5.3,
 - 13862 ◦ and the octet string m shall be the string Payload, as defined in G.5.3,
- 13863 • Otherwise if the security level, as indicated by the *gpdSecurityLevel* parameter equal to 0b10 or
13864 0b01, does not require encryption,

- 13865 ○ The octet string a shall be the string Header || Payload, as defined in G.5.3,
13866 ○ The octet string m shall be a string of length zero.

13867 The output CCM* is the string c , which consists of right-concatenation of the encrypted message Cipher text
13868 and the encrypted authentication tag U .

13869

13870 **G.5.4.2 Constructing protected GPDF**

13871 For transmission of the protected GPDF:

- 13872 • If the security level, as indicated by $gpdSecurityLevel = 0b01$:
 - 13873 ○ The fields GPD *CommandID* and GPD Command Payload remain unmodified;
 - 13874 ○ 2 LSB of U are inserted into GPDF MIC field.
 - 13875 ○ Then, the data unit is passed down using the GP-DATA.request. The MAC layer will fill
 - 13876 the MAC Sequence Number field with the value of the macDSN attribute of the MAC PIB.
 - 13877 Note: the macDSN attribute is independent of the 1LSB of the security frame counter used
 - 13878 to protect the frame.
- 13879 • Else, if the security level, as indicated by $gpdSecurityLevel = 0b10$:
 - 13880 ○ The fields GPD *CommandID* and GPD *Command Payload* remain unmodified;
 - 13881 ○ 4 LSB of U are inserted into GPDF MIC field.
 - 13882 ○ The *Frame counter* used for frame protection is inserted into GPDF Security frame counter
 - 13883 field.
- 13884 • Else if the security level, as indicated by the $gpdSecurityLevel = 0b11$:
 - 13885 ○ The *Ciphertext* is used as Payload, i.e. the *Ciphertext* replaces the fields GPD *CommandID*
 - 13886 and GPD *Command payload*;
 - 13887 ○ 4 LSB of U are inserted into GPDF MIC field;
 - 13888 ○ The *Frame counter* used for frame protection is inserted into GPDF Security frame counter
 - 13889 field.

13890 **G.5.5 Incoming frame decryption and authentication check**

13891 Determine the security level, as described in section G.5.1, and perform initialization, as described in section
13892 G.5.3.

13893 The following parameters are used for CCM* mode encryption and authentication operation, as specified in
13894 Annex A:

- 13895 • The parameter M is 4.
- 13896 • Nonce is constructed as described in section G.5.2.
- 13897 • The bit string Key determined as described in section G.4.4.

13898 If decryption is required (*SecurityLevel* 0b11), proceed with CCM* as specified in Annex A.2.3, by using
13899 *PlaintextData* = encrypted GPD *CommandID* || encrypted GPD *Command Payload* from the received GPDF.

13900 For authentication (for all *SecurityLevel* 0b01 - 0b11), calculate the U , as defined in section G.5.4.1, taking
13901 the decrypted GPD *CommandID* and GPD *Command Payload* fields as Payload, and the Header fields as
13902 defined in section G.5.3. Subsequently, compare the MIC field of the received GPDF with the corresponding
13903 number of LSB of the calculated U .

13904 Subsequently, the results are evaluated as described in section G.5.6.

13905 **G.5.6 Reporting to the next higher layer**

13906 If the authentication is successful, stub calls GP-DATA.indication with Status SECURITY_SUCCESS and
13907 carrying the unprotected GPD CommandID and GPD Command Payload.

13908 If the authentication is not successful, and

- 13909 • *SecurityLevel*=0b10 or 0b11
- 13910 • or *SecurityLevel* = 0b01 and *gppSecurityWindow* = 0,

13911 The stub calls GP-DATA.indication with Status AUTH_FAILED and carrying the protected GPD Com-
13912 mandID and GPD Command Payload.

13913 Otherwise, if the authentication is not successful and *SecurityLevel*=0b01 and if *gppSecurityWindow* pa-
13914 rameter >0, the gppSecurityWindow is decremented and Frame Counter is modified as follows: the second
13915 LSB of the Frame Counter used in the previous run is incremented by 1, and the LSB is over-written with the
13916 MAC sequence number field from the received GPDF. Then, the processing as described in section G.5.5 is
13917 performed.

13918

13919 **G.6 Inter-PAN Best Practices**

13920 Network Channel Manager Inter-PAN support is not specified in Annex E of the core stack specification
13921 ([B3]). New channel notifications will not be broadcast Inter-PAN. Inter-PAN devices which do not receive
13922 the network channel change will need to perform the network discovery procedure described in B.3.4.

13923 It is recommended that devices that use Inter-PAN should implement a whitelist of known accepted com-
13924 mands and constrain the list to only the necessary commands. Inter-PAN commands should carefully
13925 screened by the receiving device since they can be sent by devices that do not have network security cre-
13926 dentials and are performing an active attack.

13927

13928

13929

13930

13931

13932

13933

13934

13935

13936

13937

13938

13939

13940

13941

13942

13943

13944

13945

13946

13947

13948

13949

13950

13951

13952

13953

13954

13955

13956

13957

13958

13959

13960

13961

13962

This page intentionally left blank.

13963

13964

13965

13966

13967

13968

13969

13970

13971

13972

13973

13974

13975

13976

13977

ANNEX H SECURITY TEST VECTORS FOR GREEN POWER DEVICE FRAMES

H.1 Overview

The parameters marked ***bold and italics*** are dependent on device application and capabilities and thus could have other values.

Note: ‘||’ in this pseudo-code means concatenation.

All test vectors use ApplicationID = 0x00.

H.2 Security Test Vectors for a Shared Key

H.2.1 Common Settings

GP Security Key = [0xC0 , 0xC1 , 0xC2 , 0xC3 , 0xC4 , 0xC5 , 0xC6 , 0xC7 , 0xC8 , 0xC9 , 0xCA , 0xCB , 0xCC , 0Cd , 0xCe , 0Cf] = 0xCFCECDCBCAC9C8C7C6C5C4C3C2C1C0

MAC fields:

- Dest PANId = 0xffff
- Dest Addr = 0xffff
- MAC SeqNum = 0x02

NWK fields:

- NWK FC := [Ext NWK Header = 0b1 || ***Auto-Commissioning =0b0*** || ZigBee Protocol 0b0011 || Frame type =0b00] → [0b10001100] 0x8c
- GPD SrcID = 0x87654321
- Security Frame Counter = 0x00000002

Application fields:

- GPD CommandID = 0x20 (OFF)
- No data payload

H.2.2 SecurityLevel = 0b01

H.2.2.1 Transmitted Packet

Transmitted packet = MAC FC || MAC header || GP stub NWK header || Payload || MIC

14005 Transmitted test packet
14006 12 01 08 02 FF FF FF FF 8C 08 21 43 65 87 20 B7 55
14007 Note: even for SecurityLevel = 0b01, 4B MIC (U) is calculated, of which only part is transmitted in the
14008 packet.
14009

14010 **H.2.2.2 Inputs**

14011 NWK fields:
14012 Extended NWK FC = [Direction = 0b0 || RxAfterTx = 0b0 || SecurityKey = 0b0 || SecurityLevel = 0b01 ||
14013 ApplicationID = 0b000] = 0b00001000 = 0x08

14014 **H.2.2.3 Green Power Security Calculation**

14015 **Definitions**

14016 Nonce N = [0x21, 0x43, 0x65, 0x87, 0x21, 0x43, 0x65, 0x87, 0x02, 0x00, 0x00, 0x00, 0x05]
14017 a = header || Payload
14018 Header = MAC sequence number || MAC addressing fields || NWK FC || NWK_EXT FC || SrcID.

14019

14020 **Header Construction**

- 14021 1. header = 0x02 || 0xffff || 0xffff || 0x8c || 0x08 || 0x87654321
- 14022 2. header = [0x02, 0xff, 0xff, 0xff, 0xff, 0x8c, 0x08, 0x21, 0x43, 0x65, 0x87]
- 14023 3. payload = 0x20
- 14024 4. a = 0x02 || 0xffff || 0xffff || 0x8c || 0x08 || 0x87654321 || 0x20
- 14025 5. a = [0x02, 0xff, 0xff, 0xff, 0xff, 0x8c, 0x08, 0x21, 0x43, 0x65, 0x87, 0x20]

14026

14027 **Calculation**

- 14028 1. l(a) = 0x0c
- 14029 2. L(a) = 0x00 0x0c
- 14030 3. AddAuthData = L(a) || a || padding
- 14031 4. AddAuthData = [0x00, 0x0c, 0x02, 0xff, 0xff, 0xff, 0x8c, 0x08, 0x21, 0x43, 0x65, 0x87, 0x20,
14032 0x00, 0x00]
- 14033 5. Flags = [Reserved = 0b0 || Adata = 0b1 || (M-2)/2 = 0b001 || (L-1) = 0b001 → 0x49]
- 14034 6. B0 = [Flags = 0x49 || Nonce N = 0x21 0x43 0x65 0x87 0x21 0x43 0x65 0x87, 0x02, 0x00, 0x00,
14035 0x00, 0x05 || 0x00 0x00]

14036

14037 **Result**

- 14038 1. U = 0xD76F55B7
- 14039 2. MIC = 2LSB of U = 0x55B7 = [0xB7, 0x55]

14040

14041 **H.2.3 SecurityLevel = 0b10**

14042 **H.2.3.1 Transmitted Packet**

14043 Transmitted packet = MAC FC || MAC header || GP stub NWK header || Payload || MIC

14044 Transmitted test packet

14045 18 01 08 02 FF FF FF FF 8C 10 21 43 65 87 02 00 00 00 20 CF 78 7E 72

14046 **H.2.3.2 Inputs**

14047 NWK fields:

14048 NWK FC Extended = [Direction = 0b0 || RxAfterTx = 0b0 || SecurityKey = 0b0 || SecurityLevel = 0b10 ||
14049 ApplID = 0b000] → 0b00010000 → 0x10

14050

14051 **H.2.3.3 GP Security Calculation**

14052 **Definitions**

14053 Nonce N = [0x21, 0x43, 0x65, 0x87, 0x21, 0x43, 0x65, 0x87, 0x02, 0x00, 0x00, 0x00, 0x05]

14054 a = header || Payload

14055 Header = NWK FC || NWK_EXT FC || SrcID || Security Frame Counter.

14056 header = 0x8c || 0x10 || 0x87654321 || 0x00000002

14057 header = [0x8c, 0x10, 0x21, 0x43, 0x65, 0x87, 0x02, 0x00, 0x00, 0x00]

14058 payload = 0x20

14059 a = 0x8c || 0x10 || 0x87654321 || 0x00000002 || 0x20

14060 a = [0x8c, 0x10, 0x21, 0x43, 0x65, 0x87, 0x02, 0x00, 0x00, 0x00; 0x20]

14061

14062 **Calculation**

14063 1. l(a) = 0x0b

14064 2. L(a) = 0x00 0x0b

14065 3. AddAuthData = L(a) || a || padding

14066 4. AddAuthData = [0x00, 0x0b, 0x8c, 0x10, 0x21, 0x43, 0x65, 0x87, 0x02, 0x00, 0x00, 0x00, 0x20,
14067 0x00, 0x00, 0x00]

14068 5. Flags = [Reserved = 0b0 || Adata = 0b1 || (M-2)/2 = 0b001 || (L-1) = 0b001 □ 0x49]

14069 6. B0 = [Flags = 0x49 || Nonce N = 0x21 0x43 0x65 0x87 0x21 0x43 0x65 0x87, 0x02, 0x00, 0x00,
14070 0x00, 0x05 || 0x00 0x00]

14071

14072 **Result**

14073 U = 0x727E78CF

14074

MIC = FULL U = 0x727E78CF = [0xCF, 0x78, 0x7E, 0x72]

14075 **H.2.4 SecurityLevel = 0b11**

14076 **H.2.4.1 Transmitted packet**

14077 Transmitted packet = MAC FC || header || Payload || MIC

14078 Transmitted packet

14079 18 01 08 02 FF FF FF FF 8C 18 21 43 65 87 02 00 00 00 83 CA 43 24 DD

14080 **H.2.4.2 Inputs**

14081 NWK fields:

14082 NWK FC Extended = [Direction = 0b0 || RxAfterTx = 0b0 || SecurityKey = 0b0 || SecurityLevel = 0b11 ||
14083 ApplID = 0b000] → 0b00011000 → 0x18

14084

14085 **H.2.4.3 GP Security Calculation**

14086 **Definitions**

14087 Nonce N = [0x21, 0x43, 0x65, 0x87, 0x21, 0x43, 0x65, 0x87, 0x02, 0x00, 0x00, 0x00, 0x05]

14088 a = Header

14089 m = Payload

14090 Header = NWK FC || NWK_EXT FC || SrcID || Security Frame Counter.

14091 header = 0x8c || 0x18 || 0x87654321 || 0x00000002

14092 header = [0x8c, 0x18, 0x21, 0x43, 0x65, 0x87, 0x02, 0x00, 0x00, 0x00]

14093 payload = 0x20

14094 a = 0x8c || 0x18 || 0x87654321 || 0x00000002

14095 a = [0x8c, 0x18, 0x21, 0x43, 0x65, 0x87, 0x02, 0x00, 0x00, 0x00]

14096 m = 0x20

14097

14098 **Calculation**

14099 1. l(a) = 0x0a

14100 2. L(a) = 0x00 0x0a

14101 3. AddAuthData = L(a) || a || padding

14102 4. AddAuthData = [0x00, 0x0a, 0x8c, 0x18, 0x21, 0x43, 0x65, 0x87, 0x02, 0x00, 0x00, 0x00, 0x20,
14103 0x00, 0x00, 0x00]

14104 5. PlaintextData = m || padding

14105 6. PlaintextData = [0x20, 0x00,
14106 0x00, 0x00, 0x00]

14107

14108 7. AuthData = AddAuthData || PlaintextData

- 14109 8. AuthData = [0x00, 0x0a, 0x8c, 0x18, 0x21, 0x43, 0x65, 0x87, 0x02, 0x00, 0x00, 0x00, 0x20, 0x00,
14110 0x00, 0x00, 0x20, 0x00,
14111 0x00, 0x00]
- 14112 9. FlagsAuth = [Reserved = 0b0 || Adata = 0b1 || (M-2)/2 = 0b001 || (L-1) = 0b001 0x49]
- 14113 10. B0 = [FlagsAuth = 0x49 || Nonce N = 0x21 0x43 0x65 0x87 0x21 0x43 0x65 0x87, 0x02, 0x00, 0x00,
14114 0x00, 0x05 || l(m) = 0x00 0x01]
- 14115 11. B1 = [0x00, 0x0a, 0x8c, 0x18, 0x21, 0x43, 0x65, 0x87, 0x02, 0x00, 0x00, 0x00, 0x20, 0x00, 0x00,
14116 0x00]
- 14117 12. B2 = [0x20, 0x00,
14118 0x00]
- 14119 13. FlagsEncrypt = [Reserved = 0b0 || [Reserved = 0b0 || 0b000 || (L-1) = 0b001 0x01]
- 14120 14. Ai = [FlagsEncrypt = 0x01 || Nonce N = 0x21 0x43 0x65 0x87 0x21 0x43 0x65 0x87, 0x02, 0x00,
14121 0x00, 0x00, 0x05 || Counter = 0x00 0x0i]

14122

14123 Result

- 14124 U = 0xDD2443CA
14125 MIC = FULL U = 0xDD2443CA = [0xCA, 0x43, 0x24, 0xDD]
14126 Cipher = 0x83
14127

14128 **H.3 Security test vectors for an individual key**

14129 **H.3.1 Common settings**

- 14130 GP Security Key = [0xC0 , 0xC1 , 0xC2 , 0xC3 , 0xC4 , 0xC5 , 0xC6 , 0xC7 , 0xC8 , 0xC9 , 0xCA , 0xCB ,
14131 0xCC , 0xCD , 0xCE , 0xCF] = 0xCFCECDCBCAC9C8C7C6C5C4C3C2C1C0
- 14132 Nonce = 21 43 65 87 21 43 65 87 02 00 00 00 05
- 14133 MAC fields:
- 14134 • Dest PANId = 0xffff
 - 14135 • Dest Addr = 0xffff
 - 14136 • MAC SeqNum = 0x02
- 14137 NWK fields:
- 14138 • NWK FC := [Ext NWK Header = 0b1 || **Auto-Commissioning** = 0b0 || ZigBee Protocol = 0b0011 ||
14139 Frame type = 0b00] → [0b10001100] → 0x8c
 - 14140 • GPD SrcID = 0x87654321
 - 14141 • Security Frame Counter = 0x00000002
- 14142 Application fields:
- 14143
 - 14144 • GPD CommandID = **0x20 (OFF)**
 - 14145 • No data payload

14146

14147 **H.3.2 SecurityLevel=0b01**

14148 Extended NWK FC = [Direction = 0b0 || RxAfterTx = 0b0 || SecurityKey = 0b1 || SecurityLevel = 0b01 ||
14149 ApplID = 0b000] → 0x28

14150 Over the air packet:

14151 12 01 08 02 FF FF FF 8C 28 21 43 65 87 20 61 02

14152 Note: even for SecurityLevel = 0b01, 4B MIC (U) is calculated, of which only part is transmitted in the
14153 packet.

14154 **H.3.3 SecurityLevel=0b10**

14155 Extended NWK FC = [Direction = 0b0 || RxAfterTx = 0b0 || SecurityKey = 0b1 || SecurityLevel = 0b10 ||
14156 ApplID = 0b000] → 0x30

14157 Over the air packet:

14158 18 01 08 02 FF FF FF 8C 30 21 43 65 87 02 00 00 00 20 AD 69 A9 78

14159 **H.3.4 SecurityLevel=0b11**

14160 Extended NWK FC = [Direction = 0b0 || RxAfterTx = 0b0 || SecurityKey = 0b1 || SecurityLevel = 0b11 ||
14161 ApplID = 0b000] → 0x38

14162 Over the air packet:

14163 18 01 08 02 FF FF FF 8C 38 21 43 65 87 02 00 00 00 83 5F 1A 30 34

14164 **H.4 Security test vectors for bidirectional operation**

14165 **H.4.1 Common settings**

14166 For all frames:

14167 NWK Frame Type sub-field = 0b00

14168 ZigBee Protocol Version sub-field = 0b0011

14169 Auto-commissioning sub-field = 0b0

14170 Extended NWK Frame Control Present sub-field = 0b1

14171 GPD SrcID = 0x87654321

14172 Security Frame Counter = 0x44332211

14173 Security Key = { 0xC0 0xC1 0xC2 0xC3 0xC4 0xC5 0xC6 0xC7 0xC8 0xC9 0xCA 0xCB 0xCC
14174 0xCD 0xCE 0xCF }

14175 For incoming frames (from GPD to GPP / GPS):

14176 RxAfterTx sub-field = 0b1

14177 Direction sub-field = 0b0

14178 MAC Seq Nbr

14179 For SecurityLevel = 0b10 or 0b11: 0x01

14180 For SecurityLevel = 0b01: 0x11 being LSB of Security Frame Counter
14181 GPD CommandID = 0x20 (OFF)
14182 GPD Command payload = (No payload)
14183 For outgoing frames (from GPP/GPS to GPD):
14184 RxAfterTx sub-field = 0b0
14185 Direction sub-field = 0b1
14186 MAC Seq Nbr = 39
14187 GPD CommandID = 0xF3 (Channel Configuration)
14188 GPD Command payload = 0x00 (channel 11)
14189 **Note:** For SecurityLevel = 0b01: 0x11 (LSB of Security Frame Counter) is used for MIC calculation.
14190

14191 **H.4.2 Security test vectors for a shared key**

14192 For all test vectors with a shared security key:
14193 Security Key sub-field of Extended NWK Frame Control field = 0b0 (shared key)

14194 **H.4.2.1 SecurityLevel = 0b01**

14195 Incoming frame (GPD to GPP / GPS):
14196 0x12 0x01 0x08 0x11 0xFF 0xFF 0xFF 0x8C 0x48 0x21 0x43 0x65 0x87 0x20 0x16 0xB
14197 Outgoing frame (GPP/GPS to GPD):
14198 0x13 0x01 0x08 0x11 0xFF 0xFF 0xFF 0x8C 0x88 0x21 0x43 0x65 0x87 0xF3 0x00 **0x6C**
14199 **0xFD**
14200 Full 4B MIC: 0x4782**FD6C**

14201 **H.4.2.2 SecurityLevel = 0b10**

14202 Incoming frame (GPD to GPP / GPS)
14203 0x18 0x01 0x08 0x01 0xFF 0xFF 0xFF 0x8C 0x50 0x21 0x43 0x65 0x87 0x11 0x22 0x33
14204 0x44 0x20 **0xF6 0x36 0x78 0x9E**
14205 Full 4B MIC: **0x9E7836F6**
14206 Outgoing frame (GPP/GPS to GPD)
14207 0x19 0x01 0x08 0x39 0xFF 0xFF 0xFF 0x8C 0x90 0x21 0x43 0x65 0x87 0x11 0x22 0x33
14208 0x44 0xF3 0x00 **0xCC 0xA0 0xBB 0x2E**
14209 Full 4B MIC: **0x2EBBA0CC**

14210 **H.4.2.3 SecurityLevel = 0b11**

14211 Incoming frame (GPD to GPP / GPS)
14212 0x18 0x01 0x08 0x01 0xFF 0xFF 0xFF 0x8C 0x58 0x21 0x43 0x65 0x87 0x11 0x22 0x33
14213 0x44 0x2A **0x3D 0x17 0x0A 0xAA**
14214 Encrypted data: 0x2A

14215 Full 4B MIC: **0xAA0A173D**
14216 Outgoing frame (GPP/GPS to GPD)
14217 0x19 0x01 0x08 0x39 0xFF 0xFF 0xFF 0x8C 0x98 0x21 0x43 0x65 0x87 0x11 0x22 0x33
14218 0x44 0x9E 0x7E **0x14 0x0F 0xB5 0xDA**
14219 Encrypted data: 0x9E 0x7E
14220 Full 4B MIC: **0xDAB50F14**
14221

14222 **H.4.3 Security test vectors for an individual key**

14223 For all test vectors with an individual key:
14224 Security Key sub-field in NWK Ext field = 0b1 (individual key)

14225 **H.4.3.1 SecurityLevel = 0b01**

14226 Incoming frame (GPD to GPP / GPS)
14227 0x12 0x01 0x08 0x11 0xFF 0xFF 0xFF 0x8C 0x68 0x21 0x43 0x65 0x87 0x20 0x43 0x82
14228 Outgoing frame (GPP/GPS to GPD)
14229 0x13 0x01 0x08 0x11 0xFF 0xFF 0xFF 0x8C 0xA8 0x21 0x43 0x65 0x87 0xF3 0x00 **0x71**
14230 **0x15**
14231 Full 4B MIC: **0xFA601571**

14232 **H.4.3.2 SecurityLevel = 0b10**

14233 Incoming frame (GPD to GPP / GPS)
14234 0x18 0x01 0x08 0x01 0xFF 0xFF 0xFF 0x8C 0x70 0x21 0x43 0x65 0x87 0x11 0x22 0x33
14235 0x44 0x20 **0x6E 0xA9 0x51 0xBC**
14236 Full 4B MIC: **0xBC51A96E**
14237 Outgoing frame (GPP/GPS to GPD)
14238 0x19 0x01 0x08 0x39 0xFF 0xFF 0xFF 0x8C 0xB0 0x21 0x43 0x65 0x87 0x11 0x22 0x33
14239 0x44 0xF3 0x00 **0xF9 0xF1 0x7C 0x8A**
14240 Full 4B MIC: **0x8A7CF1F9**

14241 **H.4.3.3 SecurityLevel = 0b11**

14242 Incoming frame (GPD to GPP / GPS)
14243 0x18 0x01 0x08 0x01 0xFF 0xFF 0xFF 0x8C 0x78 0x21 0x43 0x65 0x87 0x11 0x22 0x33
14244 0x44 0x2A **0xD9 0xF0 0x08 0x6D**
14245 Encrypted data: 0x2A
14246 Full 4B MIC: **0x6D08F0D9**
14247 Outgoing frame (GPP/GPS to GPD)
14248 0x19 0x01 0x08 0x39 0xFF 0xFF 0xFF 0x8C 0xB8 0x21 0x43 0x65 0x87 0x11 0x22 0x33
14249 0x44 0x9E 0x7E **0xD6 0x6E 0x60 0x08**
14250 Encrypted data: 0x9E 0x7E

14251

Full 4B MIC: **0x08606ED6**

14252