



**ZigBee**<sup>®</sup>  
Control your world

# Base Device Behavior Specification

## Version 1.0

ZigBee Document 13-0402-13

February 24th, 2016

Sponsored by: ZigBee Alliance

Accepted by                      This document has been accepted for release by the ZigBee Alliance Board of Directors

Abstract                              This specification defines the base device behavior specification for devices operating on the ZigBee-PRO stack, ensuring profile interoperability between application profiles.

Keywords                              Base device, profile interoperability, ZigBee-PRO

---

Copyright © ZigBee Alliance, Inc. (1996-2016). All rights reserved.

508 Second Street, Suite 206 Davis, CA 95616 - USA

<http://www.zigbee.org>

Permission is granted to members of the ZigBee Alliance to reproduce this document for their own use or the use of other ZigBee Alliance members only, provided this notice is included. All other rights reserved. Duplication for sale, or for commercial or for-profit use is strictly prohibited without the prior written consent of the ZigBee Alliance.

1

2

This page is intentionally blank

## 3 **Notice of use and disclosure**

4 Copyright © ZigBee Alliance, Inc. (1996-2016). All rights Reserved. This  
5 information within this document is the property of the ZigBee Alliance and its use  
6 and disclosure are restricted.

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

12 No right to use any ZigBee name, logo or trademark is conferred herein. Use of any  
13 ZigBee name, logo or trademark requires membership in the ZigBee Alliance and  
14 compliance with the ZigBee Logo and Trademark Policy and related ZigBee policies.

15 This document and the information contained herein are provided on an “AS IS” basis  
16 and ZigBee **DISCLAIMS ALL WARRANTIES EXPRESS OR IMPLIED,**  
17 **INCLUDING BUT NOT LIMITED TO (A) ANY WARRANTY THAT THE USE**  
18 **OF THE INFORMATION HEREIN WILL NOT INFRINGE ANY RIGHTS OF**  
19 **THIRD PARTIES (INCLUDING WITHOUT LIMITATION ANY**  
20 **INTELLECTUAL PROPERTY RIGHTS INCLUDING PATENT, COPYRIGHT OR**  
21 **TRADEMARK RIGHTS) OR (B) ANY IMPLIED WARRANTIES OF**  
22 **MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, TITLE OR**  
23 **NONINFRINGEMENT. IN NO EVENT WILL ZIGBEE BE LIABLE FOR ANY**  
24 **LOSS OF PROFITS, LOSS OF BUSINESS, LOSS OF USE OF DATA,**  
25 **INTERRUPTION OF BUSINESS, OR FOR ANY OTHER DIRECT, INDIRECT,**  
26 **SPECIAL OR EXEMPLARY, INCIDENTAL, PUNITIVE OR CONSEQUENTIAL**  
27 **DAMAGES OF ANY KIND, IN CONTRACT OR IN TORT, IN CONNECTION**  
28 **WITH THIS DOCUMENT OR THE INFORMATION CONTAINED HEREIN,**  
29 **EVEN IF ADVISED OF THE POSSIBILITY OF SUCH LOSS OR DAMAGE. All**  
30 **Company, brand and product names may be trademarks that are the sole property of**  
31 **their respective owners.**

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

34

35

This page is intentionally blank

36

## Revision history

Revision	Date	Details	Editor
00	August 28 <sup>th</sup> , 2013	First draft	Phil Jamieson
01	October 9 <sup>th</sup> , 2013	Updates following Eindhoven workshop.	Phil Jamieson
02	November 18 <sup>th</sup> , 2013	Updates following informal review and Shanghai meeting.	Phil Jamieson
03	December 20 <sup>th</sup> , 2013	Updated with various inputs. Added some flow diagrams. Added groupcast binding mechanism.	Phil Jamieson
04	May 12 <sup>th</sup> , 2014	Updated following comments from the v0.7 super ballot.	Phil Jamieson
05	September 2 <sup>nd</sup> , 2014	Updated following comments from the v0.7 super ballot re-circulation.	Phil Jamieson
06	November 19 <sup>th</sup> , 2014	Updated following the v0.7 confirmation super ballot and PRO TSC review.	Phil Jamieson
07	April 2 <sup>nd</sup> , 2015	Updated following comments from the three proof of concept events, detailed in 15-0045.	Phil Jamieson
08	May 13 <sup>th</sup> , 2015	Updated following the Boston gated test event.	Phil Jamieson
09	August 24 <sup>th</sup> , 2015	Updated following the Hull gated test event #2.	Phil Jamieson
10	September 30 <sup>th</sup> , 2015	Further updates in preparation for the v0.9 ballot.	Phil Jamieson
11	October 30 <sup>th</sup> , 2015	Addressed comments from GTE #3 and the v0.9 ballot.	Phil Jamieson
12	December 4 <sup>th</sup> , 2015	Addressed comments from ZigBee 3.0 SVE #1.	Phil Jamieson
13	February 24 <sup>th</sup> , 2016	Addressed editorial comments from the 1.0 ballot, comments from SVE #2 and changed the document information pages.	Phil Jamieson

37

38

39

40

This page is intentionally blank

41

42	<b>Table of Contents</b>	
43	1	Introduction..... 15
44	1.1	Scope ..... 15
45	1.2	Purpose ..... 15
46	1.3	Conformance levels ..... 15
47	1.4	Conventions..... 15
48	1.4.1	Number formats ..... 15
49	1.5	Conformance testing..... 15
50	1.6	Errata ..... 16
51	2	References..... 17
52	2.1	ZigBee Alliance documents ..... 17
53	2.2	IEEE documents ..... 17
54	2.3	IETF documents ..... 17
55	3	Definitions..... 18
56	4	Acronyms and abbreviations..... 21
57	5	Environment variables ..... 22
58	5.1	Constants used by all nodes..... 22
59	5.1.1	<i>bdbcMaxSameNetworkRetryAttempts</i> constant ..... 22
60	5.1.2	<i>bdbcMinCommissioningTime</i> constant ..... 22
61	5.1.3	<i>bdbcRecSameNetworkRetryAttempts</i> constant..... 22
62	5.1.4	<i>bdbcTCLinkKeyExchangeTimeout</i> constant ..... 23
63	5.2	Constants used by nodes supporting touchlink ..... 23
64	5.2.1	<i>bdbcTLInterPANTransIdLifetime</i> constant..... 23
65	5.2.2	<i>bdbcTLMinStartupDelayTime</i> constant ..... 23
66	5.2.3	<i>bdbcTLPrimaryChannelSet</i> constant ..... 23
67	5.2.4	<i>bdbcTLRxWindowDuration</i> constant..... 24
68	5.2.5	<i>bdbcTLScanTimeBaseDuration</i> constant..... 24
69	5.2.6	<i>bdbcTLSecondaryChannelSet</i> constant..... 24
70	5.3	Attributes ..... 24
71	5.3.1	<i>bdbCommissioningGroupID</i> attribute..... 25
72	5.3.2	<i>bdbCommissioningMode</i> attribute ..... 25
73	5.3.3	<i>bdbCommissioningStatus</i> attribute..... 26

74	5.3.4	<i>bdbJoiningNodeEui64</i> attribute .....	27
75	5.3.5	<i>bdbJoiningNodeNewTCLinkKey</i> attribute .....	27
76	5.3.6	<i>bdbJoinUsesInstallCodeKey</i> attribute.....	28
77	5.3.7	<i>bdbNodeCommissioningCapability</i> attribute.....	28
78	5.3.8	<i>bdbNodeIsOnANetwork</i> attribute .....	29
79	5.3.9	<i>bdbNodeJoinLinkKeyType</i> attribute.....	30
80	5.3.10	<i>bdbPrimaryChannelSet</i> attribute .....	30
81	5.3.11	<i>bdbScanDuration</i> attribute.....	30
82	5.3.12	<i>bdbSecondaryChannelSet</i> attribute.....	30
83	5.3.13	<i>bdbTCLinkKeyExchangeAttempts</i> attribute .....	31
84	5.3.14	<i>bdbTCLinkKeyExchangeAttemptsMax</i> attribute .....	31
85	5.3.15	<i>bdbTCLinkKeyExchangeMethod</i> attribute .....	31
86	5.3.16	<i>bdbTrustCenterNodeJoinTimeout</i> attribute .....	31
87	5.3.17	<i>bdbTrustCenterRequireKeyExchange</i> attribute .....	31
88	6	General requirements .....	33
89	6.1	ZigBee logical device types .....	33
90	6.2	Network security models.....	33
91	6.3	Link keys .....	33
92	6.3.1	Default global Trust Center link key .....	34
93	6.3.2	Distributed security global link key.....	34
94	6.3.3	Install code derived preconfigured link key.....	34
95	6.3.4	Touchlink preconfigured link key.....	34
96	6.4	Use of install codes.....	34
97	6.5	Commissioning.....	35
98	6.6	Minimum requirements for all devices .....	36
99	6.7	Default reporting configuration.....	37
100	6.8	MAC data polling.....	38
101	6.9	ZigBee persistent data .....	38
102	7	Initialization .....	39
103	7.1	Initialization procedure.....	39
104	8	Commissioning .....	41
105	8.1	Top level commissioning procedure .....	41
106	8.2	Network steering procedure for a node on a network .....	43



107	8.3	Network steering procedure for a node not on a network .....	44
108	8.4	Network formation procedure .....	49
109	8.5	Finding & binding procedure for a target endpoint.....	51
110	8.6	Finding & binding procedure for an initiator endpoint .....	52
111	8.7	Touchlink procedure for an initiator .....	56
112	8.7.1	General field settings for network start/join commands .....	62
113	8.8	Touchlink procedure for a target .....	63
114	9	Reset.....	70
115	9.1	Reset via the basic cluster .....	70
116	9.2	Reset via the touchlink commissioning cluster .....	70
117	9.3	Reset via the network leave command .....	71
118	9.4	Reset via Mgmt_Leave_req ZDO command.....	71
119	9.5	Reset via a local action .....	72
120	10	Security .....	73
121	10.1	Install codes .....	73
122	10.1.1	Install code format .....	74
123	10.1.2	Hashing Function .....	75
124	10.2	Node operations.....	75
125	10.2.1	Joining node policy values.....	75
126	10.2.2	Trust Center address .....	75
127	10.2.3	Trust Center Link Keys.....	76
128	10.2.4	Requesting a Link Key.....	76
129	10.2.5	Trust Center link key exchange procedure .....	76
130	10.2.6	Receiving new Link Keys.....	80
131	10.3	Trust Center behavior .....	80
132	10.3.1	Adding the install code .....	80
133	10.3.2	Adding a new node into the network .....	81
134	10.3.3	Behavior when a known node joins .....	84
135	10.4	Distributed security network behavior .....	85
136	10.4.1	Adding a new node into the network .....	85
137	11	Annex A: Recommended practices.....	86
138	11.1	Recommendations for centralized commissioning.....	86
139	11.1.1	Centralized commissioning overview .....	86



## 145 **List of Figures**

146	Figure 1 – Initialization procedure.....	39
147	Figure 2 – Top level commissioning procedure .....	42
148	Figure 3 – Network steering procedure for a node on a network .....	44
149	Figure 4 – Network steering procedure for a node not on a network .....	46
150	Figure 5 – Network formation procedure .....	50
151	Figure 6 – Finding & binding procedure for a target endpoint.....	52
152	Figure 7 – Finding & binding procedure for an initiator endpoint .....	54
153	Figure 8 – Touchlink procedure for an initiator.....	57
154	Figure 9 – Touchlink procedure for a target .....	65
155	Figure 10 – Resetting a target to factory new via the <i>touchlink commissioning</i>	
156	cluster .....	71
157	Figure 11 – Node Install Code process .....	73
158	Figure 12 – Install code use with the Trust Center .....	74
159	Figure 13 – Trust Center link key exchange procedure sequence chart .....	77
160	Figure 14 – Trust Center link key exchange procedure.....	78
161	Figure 15 – Trust Center link key exchange procedure.....	82
162	Figure 16 – Principle of centralized commissioning with a commissioning director..	86
163		
164		

165

166

This page is intentionally blank

167

168 **List of Tables**

169	Table 1 – Constants used by all nodes .....	22
170	Table 2 – Constants used by nodes supporting touchlink .....	23
171	Table 3 – Attributes used in the base device behavior .....	24
172	Table 4 – Bits of the <i>bdbCommissioningMode</i> attribute .....	26
173	Table 5 – Values of the <i>bdbCommissioningStatus</i> attribute .....	27
174	Table 6 – Bits of the <i>bdbNodeCommissioningCapability</i> attribute .....	29
175	Table 7 – Values of the <i>bdbNodeJoinLinkKeyType</i> attribute .....	30
176	Table 8 – Values of the <i>bdbTCLinkKeyExchangeMethod</i> attribute .....	31

177

178

179  
180

This page is intentionally blank

## 181 **1 Introduction**

### 182 **1.1 Scope**

183 The scope of the base device behavior specification is to define:

- 184 • The environment required for the base device
- 185 • The initialization procedures of the base device
- 186 • The commissioning procedures of the base device
- 187 • The reset procedures of the base device
- 188 • The security procedures of the base device

189 Note: This document is intended to cover the profile interoperability technical  
190 requirements for phase 1 in relation to the base device behavior. See also [R4].

### 191 **1.2 Purpose**

192 The purpose of the base device behavior specification is to specify the environment,  
193 initialization, commissioning and operating procedures of a base device operating on  
194 the ZigBee-PRO stack to ensure profile interoperability.

### 195 **1.3 Conformance levels**

196 The key words "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT",  
197 "RECOMMENDED" and "MAY" in this document are to be interpreted as described  
198 in [R9].

### 199 **1.4 Conventions**

#### 200 **1.4.1 Number formats**

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

204 Binary numbers are specified as successive groups of 4 bits, separated by a space (" ")  
205 character from the most significant bit (next to the 0b prefix and left most on the  
206 page) to the least significant bit (rightmost on the page), e.g. the binary number  
207 0b0000 1111 represents the decimal number 15. Where individual bits are indicated  
208 (e.g. bit 3) the bit numbers are relative to the least significant bit (i.e. bit 0).

209 When a bit is specified as having a value of either 0 or 1 it is specified with an "x",  
210 e.g. "0b0000 0xxx" indicates that the lower 3 bits can take any value but the upper 5  
211 bits must each be set to 0.

### 212 **1.5 Conformance testing**

213 In order to demonstrate conformance to this specification, implementations are  
214 required to follow the appropriate test case defined in the Base Device Behavior Test  
215 Specification [R6].

**216 1.6 Errata**

217 Any errata against this specification can be found in [R7].



---

## 218 **2 References<sup>1</sup>**

### 219 **2.1 ZigBee Alliance documents**

- 220 [R1] ZigBee Specification, ZigBee Alliance document 05-3474.  
221 [R2] ZigBee Cluster Library Specification, ZigBee Alliance document 07-5123.  
222 [R3] ZigBee Application Architecture, ZigBee Alliance document 13-0589.  
223 [R4] ZigBee Profile Interoperability Technical Requirements Document, ZigBee  
224 document 13-0142-09.  
225 [R5] Installation Code Key Derivation Sample Code, ZigBee document 09-5343-04.  
226 [R6] Base Device Behavior Test Specification, ZigBee document 14-0439.  
227 [R7] Z3 Errata for Base Device Behavior 13-0402, ZigBee document 15-02020.

### 228 **2.2 IEEE documents**

- 229 [R8] Institute of Electrical and Electronics Engineers, Inc., IEEE Std. 802.15.4-2003,  
230 IEEE Standard for Information Technology —Telecommunications and  
231 Information Exchange between Systems —Local and Metropolitan Area  
232 Networks —Specific Requirements —Part 15.4: Wireless Medium Access  
233 Control (MAC) and Physical Layer (PHY) Specifications for Low Rate  
234 Wireless Personal Area Networks (WPANs). New York: IEEE Press. 2003.

### 235 **2.3 IETF documents**

- 236 [R9] S. Bradner, Key words for use in RFCs to Indicate Requirement Levels, IETF  
237 RFC 2119, March 1997.  
238

---

<sup>1</sup> The version and date information in these references was correct at the time this document was released.

### 239 **3 Definitions**

#### 240 ***Application cluster:***

241 An application cluster is a cluster that generates persistent functional transactions,  
242 e.g., a temperature measurement server cluster that reports to a client or an on/off  
243 server cluster that receives commands from a client (see also [R3]).

#### 244 ***Application transaction:***

245 An application (or functional) transaction is a cluster command, and possible  
246 response, that is generated to perform the device's persistent function, such as  
247 attribute reporting (e.g. reporting a sensor's measured value) or actuation commands  
248 (e.g. *On*, *Off*, *Toggle*, etc.). An application transaction is not a ZDO transaction, one-  
249 time transaction, or commissioning transaction.

250 The cluster that generates the application transaction is the initiator. A corresponding  
251 cluster that receives the initial message of the transaction is the target. The same  
252 cluster on multiple endpoints/nodes could be the target of an application transaction,  
253 because of multiple source bindings or bindings with a group or broadcast destination.

#### 254 ***Bind or binding (verb):***

255 Create a binding or the act of creating a binding.

#### 256 ***Binding (noun):***

257 A binding is a ZigBee source binding table entry on a node which indicates where  
258 data is sent to from a cluster on an endpoint (see also [R3]).

#### 259 ***Centralized security network:***

260 A centralized security network is a ZigBee network formed by a ZigBee coordinator  
261 with the functionality of a Trust Center. Each node that joins such a network is  
262 authenticated with the Trust Center before it can operate on the network.

#### 263 ***Commissioning director:***

264 A node in a network that is able to directly edit bindings and reporting configurations  
265 on any node in the network.

#### 266 ***Device:***

267 An application implementation corresponding to a ZigBee defined device type with a  
268 unique device identifier and part of a node. A device is resident on a single endpoint,  
269 called a device endpoint. A single node can have one or more devices (see also [R3]).

#### 270 ***Distributed security network:***

271 A distributed security network is a ZigBee network formed by a ZigBee router and  
272 which does not have a Trust Center. Each node that joins such a network is  
273 authenticated by its parent before it can operate on the network.

#### 274 ***Dynamic device:***

275 A dynamic device is an application implementation of an endpoint that has no specific  
276 set of application clusters (see also [R3]).

**277 EZ-Mode:**

278 EZ-Mode is a commissioning method that defines network steering and device reset  
279 on the node as well as finding & binding for endpoints with target or initiator clusters.  
280 The method requires that a product supports interactive mechanisms to invoke the  
281 method. These mechanisms are accessible to the installer of the product. These  
282 mechanisms are implementation dependent and can be overloaded and/or automatic.

283 Invoking EZ-Mode on a device endpoint puts the node and device in EZ-Mode for 3 a  
284 minute window. Each time EZ-Mode is invoked on a device, it extends the window  
285 for another 3 minutes. During the window, nodes perform EZ-Mode Network  
286 Steering and devices perform EZ-Mode Finding & Binding to other devices in EZ-  
287 Mode. Target devices use the Identify cluster to identify during the window. Initiator  
288 devices actively discover targets during the window and then bind to corresponding  
289 target clusters.

**290 EZ-Mode finding & binding:**

291 EZ-Mode finding & binding is the process of automatically establishing application  
292 connections, by using the identify cluster, between matching application clusters on  
293 two or more devices (see also [R3]). Note that hereafter “EZ-Mode finding &  
294 binding” is referred to as “finding & binding”.

**295 EZ-Mode network steering:**

296 For a node that is not already joined to a network, EZ-Mode network steering is the  
297 action of searching for and joining an open network. For a node that has joined a  
298 network, EZ-Mode network steering is the action of opening the network to allow  
299 new nodes to join. Note that hereafter “EZ-Mode network steering” is referred to as  
300 “network steering”.

**301 Finding & binding:**

302 See *EZ-Mode finding & binding*.

**303 Initiator cluster:**

304 An initiator cluster is an application cluster that initiates cluster transactions (see also  
305 [R3]).

**306 Joined:**

307 A node is said to be joined to a network if it has successfully executed the network  
308 joining process or has formed a network. Note that if the node formed the network it  
309 is possible that it does not yet have any peer nodes with which to communicate.  
310 Similarly, if a node has joined a network it is possible that it does not yet have any  
311 bound endpoints.

**312 Network steering:**

313 See *EZ-Mode network steering*.

**314 Node:**

315 A node defines a single instance of the ZigBee-PRO stack with a single IEEE address  
316 on a single network. A node is made up of one or more logical device instances each

317 represented on an endpoint and a node can have a node endpoint which is an instance  
318 for the entire node, e.g., the ZDO on endpoint 0 (see also [R3]).

319 ***Simple device:***

320 A simple device is an application implementation of an application specific endpoint  
321 that has mandatory application clusters (see also [R3]).

322 ***Target cluster:***

323 A target cluster is an application cluster that receives the initiated messages from an  
324 initiator cluster and could potentially respond to the initiator (see also [R3]).

325 ***Touchlink commissioning:***

326 Touchlink commissioning is an optional commissioning mechanism where nodes are  
327 commissioned on a network using commands sent using inter-PAN communication in  
328 close physical proximity.

329 ***Utility cluster:***

330 A utility cluster is a cluster whose function is not part of the persistent functional  
331 operation of the product. Function examples: commissioning, configuration,  
332 discovery, etc.

333 ***ZigBee coordinator:***

334 A ZigBee coordinator is a ZigBee logical device type that includes the functionality  
335 of a Trust Center and is responsible for starting a centralized security network and  
336 managing node joining and key distribution for the network. A ZigBee coordinator  
337 has the *logical type* field of the node descriptor set to 0b000.

338 ***ZigBee end device:***

339 A ZigBee end device is a ZigBee logical device type that can only join an existing  
340 network. A ZigBee end device has the *logical type* field of the node descriptor set to  
341 0b010.

342 ***ZigBee router:***

343 A ZigBee router is a ZigBee logical device type that is responsible for managing node  
344 joining. A ZigBee router cannot start a centralized security network but it can start a  
345 distributed security network. A ZigBee router has the *logical type* field of the node  
346 descriptor set to 0b001.

## 347 **4 Acronyms and abbreviations**

348	<b>AES</b>	Advanced Encryption Standard
349	<b>AIB</b>	Application support sub-layer information base
350	<b>APS</b>	Application support sub-layer
351	<b>APSME</b>	Application support sub-layer management entity
352	<b>CBKE</b>	Certificate based key exchange
353	<b>CCITT</b>	Comité Consultatif International Téléphonique et Télégraphique
354	<b>CD</b>	Commissioning director
355	<b>CRC</b>	Cyclic redundancy check
356	<b>EP</b>	Endpoint
357	<b>EUI</b>	Extended unique identifier
358	<b>ID</b>	Identifier
359	<b>IEEE</b>	Institute of Electrical and Electronic Engineers
360	<b>LQI</b>	Link quality indication
361	<b>MAC</b>	Medium access control
362	<b>MMO</b>	Matyas-Meyer-Oseas
363	<b>NLME</b>	Network layer management entity
364	<b>NVRAM</b>	Non-volatile random access memory
365	<b>NWK</b>	Network
366	<b>OTA</b>	Over the air
367	<b>PAN</b>	Personal area network
368	<b>PHY</b>	Physical
369	<b>TC</b>	Trust Center
370	<b>WPAN</b>	Wireless personal area network
371	<b>ZC</b>	ZigBee coordinator
372	<b>ZCL</b>	ZigBee cluster library
373	<b>ZDO</b>	ZigBee device objects
374	<b>ZED</b>	ZigBee end device
375	<b>ZR</b>	ZigBee router
376		

## 377 **5 Environment variables**

378 This clause specifies the constants and attributes required to implement a node  
379 conforming to the base device behavior specification.

380 All constants specified in this specification use the prefix “*bdbc*” (*base device*  
381 *behavior constant*) and all attributes use the prefix “*bdb*” (*base device behavior*).

### 382 **5.1 Constants used by all nodes**

383 Table 1 lists the set of constants defined by the base device behavior specification that  
384 are used by all devices.

385  
386 **Table 1 – Constants used by all nodes**

Constant	Value
<i>bdbcMaxSameNetworkRetryAttempts</i>	10
<i>bdbcMinCommissioningTime</i>	180s (0xb4)
<i>bdbcRecSameNetworkRetryAttempts</i>	3
<i>bdbcTCLinkKeyExchangeTimeout</i>	5s

387

#### 388 **5.1.1 *bdbcMaxSameNetworkRetryAttempts* constant**

389 The *bdbcMaxSameNetworkRetryAttempts* constant specifies the maximum number of  
390 join or key exchange attempts made to the same network.

391 This constant is used by each node.

392 See also *bdbcRecSameNetworkRetryAttempts*.

#### 393 **5.1.2 *bdbcMinCommissioningTime* constant**

394 The *bdbcMinCommissioningTime* constant specifies the minimum duration in seconds  
395 for which a network is opened to allow new nodes to join or for a device to identify  
396 itself.

397 This constant is used by each node.

#### 398 **5.1.3 *bdbcRecSameNetworkRetryAttempts* constant**

399 The *bdbcRecSameNetworkRetryAttempts* constant specifies the RECOMMENDED  
400 maximum number of join or key exchange attempts made to the same network.

401 This constant is used by each node.

402 See also *bdbcMaxSameNetworkRetryAttempts*.

#### 403 **5.1.4 *bdbcTCLinkKeyExchangeTimeout* constant**

404 The *bdbcTCLinkKeyExchangeTimeout* constant specifies the maximum time in  
405 seconds a joining node will wait for a response when sending an APS request key to  
406 the Trust Center.

407 This constant is used by each node.

### 408 **5.2 Constants used by nodes supporting touchlink**

409 Table 2 lists the set of constants defined by the base device behavior specification that  
410 are used by those devices which support touchlink commissioning.

411

412 **Table 2 – Constants used by nodes supporting touchlink**

Constant	Value
<i>bdbcTLInterPANTransIdLifetime</i>	8s
<i>bdbcTLMinStartupDelayTime</i>	2s
<i>bdbcTLPrimaryChannelSet</i>	0x02108800
<i>bdbcTLRxWindowDuration</i>	5s
<i>bdbcTLScanTimeBaseDuration</i>	0.25s
<i>bdbcTLSecondaryChannelSet</i>	0x07ff800 XOR <i>bdbcTLPrimaryChannelSet</i>

413

#### 414 **5.2.1 *bdbcTLInterPANTransIdLifetime* constant**

415 The *bdbcTLInterPANTransIdLifetime* constant specifies the maximum length of time an  
416 inter-PAN transaction ID remains valid.

417 This constant is used by a node if touchlink is supported.

#### 418 **5.2.2 *bdbcTLMinStartupDelayTime* constant**

419 The *bdbcTLMinStartupDelayTime* constant specifies the length of time an initiator  
420 waits to ensure the target has completed its network startup procedure.

421 This constant is used by a node if touchlink is supported.

#### 422 **5.2.3 *bdbcTLPrimaryChannelSet* constant**

423 The *bdbcTLPrimaryChannelSet* constant specifies the bitmask for the channel set  
424 comprised of channels 11, 15, 20 and 25, that will be used for a non-extended  
425 touchlink scan.

426 This constant is used by a node if touchlink is supported.

#### 427 **5.2.4 *bdbcTLRxWindowDuration* constant**

428 The *bdbcTLRxWindowDuration* constant specifies the maximum duration that a node  
429 leaves its receiver enabled during touchlink for subsequent responses.

430 This constant is used by a node if touchlink is supported.

#### 431 **5.2.5 *bdbcTLScanTimeBaseDuration* constant**

432 The *bdbcTLScanTimeBaseDuration* constant specifies the base duration for a  
433 touchlink scan operation during which the receiver is enabled for scan responses after  
434 having transmitted a scan request.

435 This constant is used by a node if touchlink is supported.

#### 436 **5.2.6 *bdbcTLSecondaryChannelSet* constant**

437 The *bdbcTLSecondaryChannelSet* constant specifies the bitmask for the channel set  
438 comprised of the remaining IEEE 802.15.4-2003 channels available at 2.4GHz that  
439 will be used for an extended touchlink scan after the *bdbcTLPrimaryChannelSet*  
440 channels have been scanned.

441 This constant is used by a node if touchlink is supported.

### 442 **5.3 Attributes**

443 The base device behavior specification defines the set of attributes listed in Table 3.  
444 The “Used by” column indicates for which ZigBee logical device type the attribute is  
445 used and whether the attribute is to be defined per endpoint. Note: all attributes  
446 defined in this specification are internal to the node and not available over air.

447

448 **Table 3 – Attributes used in the base device behavior**

Attribute	Data type	Range	Default value	Used by
<i>bdbCommissioningGroupID</i>	Unsigned 16-bit integer	0x0001 – 0xffff	0xffff	Initiator nodes, per endpoint
<i>bdbCommissioningMode</i>	8-bit bitmap	0b0000 xxxx	0b0000 0000	All nodes, per endpoint
<i>bdbCommissioningStatus</i>	Enumeration	See Table 5	SUCCESS	All nodes, per endpoint
<i>bdbJoiningNodeEui64</i>	IEEE Address	Any value within the range of the data type	All zero (invalid address)	ZC
<i>bdbJoiningNodeNewTCLinkKey</i>	128-bit security key	Any value within the range of the data type	All zero (invalid key value)	ZC
<i>bdbJoinUsesInstallCodeKey</i>	Boolean	TRUE or FALSE	FALSE	ZC
<i>bdbNodeCommissioning-Capability</i>	8-bit bitmap	0b0000 xxx1	0b0000 0001	All nodes
<i>bdbNodeIsOnANetwork</i>	Boolean	TRUE or FALSE	FALSE	All nodes



Attribute	Data type	Range	Default value	Used by
<i>bdbNodeJoinLinkKeyType</i>	Unsigned 8-bit integer	0x00 – 0x02	0x00	ZR, ZED
<i>bdbPrimaryChannelSet</i>	32-bit bitmap	0x00000800 – 0x07fff800	0x02108800	All nodes
<i>bdbScanDuration</i>	Unsigned 8-bit integer	0x00 – 0x0e	0x04	All nodes
<i>bdbSecondaryChannelSet</i>	32-bit bitmap	0x00000800 – 0x07fff800	0x07fff800 XOR <i>bdbPrimaryChannelSet</i>	All nodes
<i>bdbTCLinkKeyExchange-Attempts</i>	Unsigned 8-bit integer	0x00 – 0xff	0x00	ZR, ZED
<i>bdbTCLinkKeyExchange-AttemptsMax</i>	Unsigned 8-bit integer	0x00 – 0xff	0x03	ZR, ZED
<i>bdbTCLinkKeyExchange-Method</i>	Unsigned 8-bit integer	0x00 – 0x01 (0x02 – 0xff are reserved)	0x00	ZR, ZED
<i>bdbTrustCenterNodeJoin-Timeout</i>	Unsigned 8-bit integer	0x00 – 0xff	0x0f (seconds)	ZC
<i>bdbTrustCenterRequireKey-Exchange</i>	Boolean	TRUE or FALSE	TRUE	ZC

449

### 450 **5.3.1 *bdbCommissioningGroupID* attribute**

451 The *bdbCommissioningGroupID* attribute specifies the identifier of the group on  
452 which the initiator applies finding & binding. If *bdbCommissioningGroupID* is equal  
453 to 0xffff, any bindings will be created as unicast.

454 This attribute is only used during commissioning if bit 3 of the  
455 *bdbCommissioningMode* attribute (see sub-clause 5.3.2) is equal to 1 (finding &  
456 binding is to be attempted).

457 This attribute is used by initiator nodes, per endpoint.

458 Note: sleeping ZigBee end device targets will not be able to benefit from groupcast  
459 transmissions (see the *groups* cluster in [R2] for more details).

### 460 **5.3.2 *bdbCommissioningMode* attribute**

461 The *bdbCommissioningMode* attribute is used as a parameter to the top level  
462 commissioning procedure and specifies the commissioning methods and options taken  
463 when commissioning is invoked, represented by each bit from the least significant bit  
464 to the most significant bit.

465 Note that this attribute is different to the *bdbNodeCommissioningCapability* attribute  
466 which specifies which commissioning mechanisms are supported by the node. The  
467 attribute is a bitwise or of the bits listed in Table 4.

468 This attribute is used by all nodes, per endpoint.

469

470

**Table 4 – Bits of the *bdbCommissioningMode* attribute**

<i>bdbCommissioning-Mode</i> bit	Description
0	Touchlink: 0 = Do not attempt Touchlink commissioning 1 = Attempt Touchlink commissioning
1	Network steering: 0 = Do not attempt network steering 1 = Attempt network steering
2	Network formation: 0 = Do not attempt to form a network 1 = Attempt to form a network, according to device type <sup>2</sup>
3	Finding & binding: 0 = Do not attempt finding & binding 1 = Attempt finding & binding
4-7	Reserved (set to zero)

471

**472 5.3.3 *bdbCommissioningStatus* attribute**

473 The *bdbCommissioningStatus* attribute specifies the status of its commissioning  
474 attempt and can be set to one of the values listed in Table 5.

475 This attribute is used by all nodes, per endpoint.

476

---

<sup>2</sup> If the device is a ZigBee coordinator (Trust Center), then this bit indicates that the device will form a centralized security network. If the device is a ZigBee router, then this bit indicates that the device will form a distributed security network.

477

**Table 5 – Values of the *bdbCommissioningStatus* attribute**

<b>Value of the <i>bdbCommissioningStatus</i> attribute</b>	<b>Description</b>
SUCCESS	The commissioning sub-procedure was successful.
IN_PROGRESS	One of the commissioning sub-procedures has started but is not yet complete.
NOT_AA_CAPABLE	The initiator is not address assignment capable during touchlink.
NO_NETWORK	A network has not been found during network steering or touchlink.
TARGET_FAILURE	A node has not joined a network when requested during touchlink.
FORMATION_FAILURE	A network could not be formed during network formation.
NO_IDENTIFY_QUERY_RESPONSE	No response to an <i>identify query</i> command has been received during finding & binding.
BINDING_TABLE_FULL	A binding table entry could not be created due to insufficient space in the binding table during finding & binding.
NO_SCAN_RESPONSE	No response to a <i>scan request</i> inter-PAN command has been received during touchlink.
NOT_PERMITTED	A touchlink (steal) attempt was made when a node is already connected to a centralized security network.
TCLK_EX_FAILURE	The Trust Center link key exchange procedure has failed attempting to join a centralized security network.

478

**479 5.3.4 *bdbJoiningNodeEui64* attribute**

480 The *bdbJoiningNodeEui64* attribute contains the EUI-64 of the node joining the  
481 centralized security network.

482 This attribute is used by ZigBee coordinator nodes.

**483 5.3.5 *bdbJoiningNodeNewTCLinkKey* attribute**

484 The *bdbJoiningNodeNewTCLinkKey* attribute contains the new link key established with the  
485 joining node but which has not yet been confirmed.

486 This attribute is used by ZigBee coordinator nodes.

487 **5.3.6 *bdbJoinUsesInstallCodeKey* attribute**

488 The *bdbJoinUsesInstallCodeKey* attribute specifies the Trust Center's policy that  
489 indicates whether it requires an install code derived preconfigured link key to be  
490 preinstalled before the corresponding node joins its network.

491 If *bdbJoinUsesInstallCodeKey* is equal to FALSE, the Trust Center permits a node to  
492 join its network without having a corresponding install code derived preconfigured  
493 link key associated with the node preinstalled before the node joins. If

494 *bdbJoinUsesInstallCodeKey* is equal to TRUE, the Trust Center only permits a node  
495 to join its network if a corresponding install code derived preconfigured link key  
496 associated with the node has been preinstalled before the node joins.

497 This attribute is used by ZigBee coordinator nodes.

498 **5.3.7 *bdbNodeCommissioningCapability* attribute**

499 The *bdbNodeCommissioningCapability* attribute specifies the commissioning  
500 capabilities of the node. The attribute is a bitwise or of the bits listed in Table 6.

501 This attribute is used by all nodes.

502

**Table 6 – Bits of the *bdbNodeCommissioningCapability* attribute**

<i>bdbCommissioning-Capability</i> bit	Description
0	Network steering: 0 = Forbidden 1 = The node supports network steering  All nodes set this bit to 1, indicating mandatory support for network steering.
1	Network formation: 0 = The node will not form a network 1 = The node will form a network, according to ZigBee logical device type  ZigBee coordinator (Trust Center) nodes set this bit to 1, indicating that it will always form a centralized security network.
2	Finding & binding: 0 = The node does not contain any device endpoints for which finding & binding is mandated 1 = The node contains device endpoints in which finding & binding is mandated  This bit is set according to the specific devices implemented on the node. If a simple device is implemented, this bit is set to 1. If only a dynamic device is implemented, this bit is set to 1 if finding & binding is supported on that device.
3	Touchlink commissioning: 0 = The node does not support Touchlink commissioning 1 = The node supports Touchlink commissioning
4-7	Reserved (set to zero)

503

**504 5.3.8 *bdbNodeIsOnANetwork* attribute**

505 The *bdbNodeIsOnANetwork* attribute indicates whether a node is joined to a network.  
506 If *bdbNodeIsOnANetwork* is equal to FALSE, the node has not yet formed or joined a  
507 network. If *bdbNodeIsOnANetwork* is equal to TRUE, the node has either formed a  
508 centralized security network (if the node is a ZigBee coordinator), formed a  
509 distributed security network (if the node is a ZigBee router) or has joined a network  
510 (if the node is a ZigBee router or a ZigBee end device). Note that when  
511 *bdbNodeIsOnANetwork* is equal to TRUE, it is possible for the node to not yet have  
512 any bound endpoints.

513 This attribute is used by all nodes.

### 514 **5.3.9 *bdbNodeJoinLinkKeyType* attribute**

515 The *bdbNodeJoinLinkKeyType* attribute indicates the type of link key (see sub-clause  
516 6.3) with which the node was able to decrypt the network key when the node joins a  
517 new network. This attribute can take one of the values listed in Table 7.

518

519 **Table 7 – Values of the *bdbNodeJoinLinkKeyType* attribute**

Value of the <i>bdbNodeJoinLinkKeyType</i> attribute	Network model	Type of link key
0x00	Centralized	Default global Trust Center link key
0x01	Distributed	Distributed security global link key
0x02	Centralized	Install code derived preconfigured link key
0x03	Distributed	Touchlink preconfigured link key

520

521 This attribute is used by all ZigBee router and ZigBee end device nodes.

### 522 **5.3.10 *bdbPrimaryChannelSet* attribute**

523 The *bdbPrimaryChannelSet* attribute specifies the channel set, defined by the  
524 application, that will be used in preference, e.g. during a channel scan. Note that if a  
525 primary scan is not required, this attribute is set to 0x00000000. However, in this  
526 case, *bdbSecondaryChannelSet* is not to be set to 0x00000000.

527 This attribute is used by all nodes.

### 528 **5.3.11 *bdbScanDuration* attribute**

529 The *bdbScanDuration* attribute specifies the duration of an IEEE 802.15.4 scan  
530 operation per channel. The time spent scanning each channel is given by  
531  $[aBaseSuperframeDuration * (2^n + 1)]$ , where  $n$  is the value of *bdbScanDuration* and  
532 *aBaseSuperframeDuration* is defined in sub-clause 7.4.1 (Table 70) of [R8].

533 The scan is performed indirectly via the ZigBee primitives and can be energy, passive  
534 or active.

535 This attribute is used by all nodes.

### 536 **5.3.12 *bdbSecondaryChannelSet* attribute**

537 The *bdbSecondaryChannelSet* attribute specifies the channel set, defined by the  
538 application, that will be used after the primary channels, e.g. during a channel scan.  
539 Note that if a secondary scan is not required, this attribute is set to 0x00000000.  
540 However, in this case, *bdbPrimaryChannelSet* is not to be set to 0x00000000.

541 This attribute is used by all nodes.

542 **5.3.13 *bdbTCLinkKeyExchangeAttempts* attribute**

543 The *bdbTCLinkKeyExchangeAttempts* attribute contains the number of key  
544 establishment attempts that have been made to establish a new link key after joining.

545 This attribute is used by all ZigBee router and ZigBee end device nodes.

546 **5.3.14 *bdbTCLinkKeyExchangeAttemptsMax* attribute**

547 The *bdbTCLinkKeyExchangeAttemptsMax* attribute specifies the maximum number of  
548 key establishment attempts that will be made before giving up on the key  
549 establishment.

550 This attribute is used by all ZigBee router and ZigBee end device nodes.

551 **5.3.15 *bdbTCLinkKeyExchangeMethod* attribute**

552 The *bdbTCLinkKeyExchangeMethod* attribute specifies the method used to establish a  
553 new link key after joining the network and can be set to one of the non-reserved  
554 values listed in Table 8.

555 This attribute is used by all ZigBee router and ZigBee end device nodes.

556

557 **Table 8 – Values of the *bdbTCLinkKeyExchangeMethod* attribute**

Value of the <i>bdbTCLinkKeyExchangeMethod</i> attribute	Description
0x00	APS Request Key
0x01	Certificate Based Key Exchange (CBKE)
0x02 – 0xff	Reserved

558

559 **5.3.16 *bdbTrustCenterNodeJoinTimeout* attribute**

560 The *bdbTrustCenterNodeJoinTimeout* attribute specifies a timeout in seconds for the  
561 Trust Center to remove the Trust Center link key of the newly joined node that did not  
562 successfully establish a new link key.

563 This attribute is used by ZigBee coordinator nodes.

564 **5.3.17 *bdbTrustCenterRequireKeyExchange* attribute**

565 The *bdbTrustCenterRequireKeyExchange* attribute specifies whether the Trust Center  
566 requires a joining device to exchange its initial link key with a new link key generated  
567 by the Trust Center. If *bdbTrustCenterRequireKeyExchange* is equal to TRUE, the  
568 joining node must undergo the link key exchange procedure; failure to exchange the  
569 link key will result in the node being removed from the network. If

- 570 *bdbTrustCenterRequireKeyExchange* is equal to FALSE, the Trust Center will permit  
571 the joining node to remain on the network without exchanging its initial link key.  
572 This attribute is used by ZigBee coordinator nodes.



## 573 **6 General requirements**

574 This clause specifies the general requirements for all nodes implementing the base  
575 device behavior specification.

### 576 **6.1 ZigBee logical device types**

577 A node designated as having a logical device type of a ZigBee coordinator SHALL  
578 also encompass the role of the Trust Center. A ZigBee coordinator SHALL form a  
579 centralized security network and, as such, SHALL NOT attempt to join another  
580 network.

581 A node designated as having a logical device type of a ZigBee router SHALL be able  
582 to join an existing centralized or distributed security network. However, a ZigBee  
583 router SHALL NOT form a centralized security network but MAY form a distributed  
584 security network if an existing centralized or distributed security network is not  
585 available to join.

586 A node designated as having a logical device type of a ZigBee end device SHALL be  
587 able to join an existing centralized or distributed security network.

588 A node MAY support the capability of being both a ZigBee coordinator and a ZigBee  
589 router, switchable under application control. However, at any one time, the node  
590 SHALL be designated as being one type or the other. This allows the scenario of a  
591 node trying to join a network as a ZigBee router and if there are no networks to join,  
592 the node can switch to being a ZigBee coordinator and, as a result, form a centralized  
593 security network. Once the node has formed or joined a network, it SHALL NOT  
594 change its type unless it first destroys or leaves, respectively, that network.

### 595 **6.2 Network security models**

596 A ZigBee network MAY support a centralized security model (a centralized security  
597 network) or a distributed security model (a distributed security network). All none  
598 ZigBee coordinator nodes SHALL be able to join a network supporting either model  
599 and adapt to the security conditions of the network they are joining (see sub-clause  
600 4.6.3 of [R1]). This adaptation SHOULD be as seamless as possible to the user.

### 601 **6.3 Link keys**

602 Each node SHALL contain the following link keys:

- 603 1. The default global Trust Center link key
- 604 2. The distributed security global link key
- 605 3. An install code derived preconfigured link key

606 In addition, if a node supports touchlink commissioning, it SHALL also contain the  
607 following link key:

- 608 4. The touchlink preconfigured link key

609 The *bdbNodeJoinLinkKeyType* attribute indicates the type of link key that was used to  
610 decrypt the network key during joining.

### 611 **6.3.1 Default global Trust Center link key**

612 The default global Trust Center link key is a link key that is supported by all devices  
 613 and can be used to join a centralized security network if no other link key is specified.  
 614 This link key SHALL have a value of:

		0x5a	0x69	0x67	0x42
Default global Trust Center	=	0x65	0x65	0x41	0x6c
link key (0:15)		0x6c	0x69	0x61	0x6e
		0x63	0x65	0x30	0x39

615

### 616 **6.3.2 Distributed security global link key**

617 The distributed security global link key is used to join a distributed security network.  
 618 This link key is provided to a company as a result of a successful certification of a  
 619 product. For testing, this key SHALL have the value of:

		0xd0	0xd1	0xd2	0xd3
Distributed security global	=	0xd4	0xd5	0xd6	0xd7
link key (0:15)		0xd8	0xd9	0xda	0xdb
		0xdc	0xdd	0xde	0xdf

620

### 621 **6.3.3 Install code derived preconfigured link key**

622 The install code derived preconfigured link key is generated from a random install  
 623 code created for the product and provided to the node in a manufacturer-specific way  
 624 and referred to during installation. See sub-clause 10.1 for more details.

### 625 **6.3.4 Touchlink preconfigured link key**

626 The touchlink preconfigured link key is used to join a network via touchlink. This  
 627 link key is provided to a company as a result of a successful certification of a product.  
 628 For testing, this key SHALL have the value of:

		0xc0	0xc1	0xc2	0xc3
Touchlink preconfigured	=	0xc4	0xc5	0xc6	0xc7
link key (0:15)		0xc8	0xc9	0xca	0xcb
		0xcc	0xcd	0xce	0xcf

629 A node using the touchlink preconfigured link key in the touchlink procedure SHALL  
 630 set either bit 4 or bit 15 of the *key bitmask* field of the *scan response* inter-PAN  
 631 command frame to 1 (see [R2]), depending on whether the node is being used during  
 632 certification testing or in post-certification production use (normal operation),  
 633 respectively.

## 634 **6.4 Use of install codes**

635 All nodes SHALL support install codes.

636 Nodes that are not available via retail channels and that are professionally installed  
637 (e.g., an electricity or gas meter) MAY be configured to require the use of install  
638 codes on joining.

639 Nodes that are available via retail channels and that support a user configuration  
640 mechanism (e.g., a physical switch) MAY default to a mode in which only networks  
641 that require the use of install codes for joining are considered. However, there  
642 SHALL be a mechanism to switch into a mode in which all networks are considered  
643 for joining.

644 Nodes that are available via retail channels but do not have a user configuration  
645 mechanism SHALL be able to join all networks automatically.

646 The Trust Center MAY require the use of install codes for all nodes joining its  
647 network.

## 648 **6.5 Commissioning**

649 All nodes SHALL support network steering so that a common mechanism can be used  
650 as a fall back by all nodes. Devices implementing a simple device class SHALL  
651 support finding & binding whereas devices implementing either a dynamic or a node  
652 device class MAY support finding & binding. Other commissioning mechanisms  
653 MAY be supported according to the individual device specifications implemented on  
654 the node.

655 The commissioning mechanisms that are supported by a node are specified in the  
656 *bdbNodeCommissioningCapability* attribute (see sub-clause 5.3).

657 This specification specifies the procedures for the following commissioning  
658 mechanisms:

- 659 • Network steering. All nodes SHALL support network steering.
- 660 • Network formation. The ability of a node to form a network and its network  
661 security model SHALL be dependent on the logical device type of the node.
- 662 • Finding & binding. The ability to locate and bind to application clusters on  
663 other devices SHALL be supported on devices implementing a simple device  
664 class and MAY be supported on devices implementing either a dynamic or a  
665 node device class.
- 666 • Touchlink commissioning. A node MAY support the proximity based  
667 commissioning mechanism. If touchlink commissioning is supported, the  
668 node SHALL support touchlink as an initiator, a target or both.

669 An implementation MAY use commissioning at any time so, for example, network  
670 steering can be performed at any time for the whole node or finding & binding can be  
671 performed at any time on any endpoint appropriate to the application. However, each  
672 time it is used it SHALL be executed as specified in the top-level commissioning  
673 procedure.

674 For example, a node which implements a temperature sensor device on a single  
675 endpoint can use the commissioning procedure on the activation of a specific user  
676 button press. Similarly, a node which implements an on/off light switch device on

677 two endpoints (one for each switch) can use the commissioning procedure on  
678 activation of each switch.

679 The required commissioning procedure is controlled by a number of attributes that are  
680 defined per active endpoint (see also sub-clause 5.3): *bdbCommissioningMode*,  
681 *bdbCommissioningGroupID* and *bdbCommissioningStatus*. To execute  
682 commissioning, the required commissioning options to execute at that time are  
683 specified in the appropriate *bdbCommissioningMode* attribute. If finding & binding is  
684 required, the *bdbCommissioningGroupID* (the group to use for the finding & binding)  
685 is also specified. Note that if a group binding is not required, the  
686 *bdbCommissioningGroupID* attribute is set to 0xffff. After the requested  
687 commissioning options are executed, the *bdbCommissioningStatus* attribute indicates  
688 the status of the attempt.

689 The commissioning options specified in *bdbCommissioningMode* are executed in the  
690 order least significant bit first, i.e., touchlink commissioning first, then network  
691 steering, then network formation and finally finding & binding, as follows:

- 692 1. If touchlink commissioning as an initiator is specified and it is successful, no  
693 further commissioning options specified in *bdbCommissioningMode* SHALL  
694 be executed during that invocation of the commissioning procedure. Note that  
695 touchlink is deemed to be successful if a response to a touchlink scan request  
696 is received by the initiator.
- 697 2. If network steering is specified, the node SHALL attempt network steering  
698 according to whether the node is joined to a network or not.
- 699 3. If network formation is specified the node SHALL only attempt network  
700 formation if the node is not yet joined to a network. As such, if network  
701 steering is specified and it is successful, then the node SHALL NOT attempt  
702 network formation. If network formation is specified and the node is a ZigBee  
703 coordinator it SHALL attempt to form a centralized security network.  
704 Conversely, if network formation is specified and the node is a ZigBee router  
705 it SHALL attempt to form a distributed security network. If the node is a  
706 ZigBee end device it SHALL skip network formation.
- 707 4. If finding & binding is specified the node SHALL only attempt finding &  
708 binding if it is operational on a network. Finding & binding MAY be  
709 instigated on one or more of the endpoints implemented on a node and its form  
710 is dependent on the cluster class (see [R3] for details). For a type 1 client or a  
711 type 2 server cluster, the application SHALL perform finding & binding as an  
712 initiator endpoint. Conversely, for a type 1 server or type 2 client cluster, the  
713 application SHALL perform finding & binding as a target endpoint.

## 714 **6.6 Minimum requirements for all devices**

715 All nodes SHALL support the following requirements:

- 716 • A node SHALL process the ZDO discovery service commands:  
717 *Active\_EP\_req*, *Node\_Desc\_req*, *Simple\_Desc\_req*, *IEEE\_addr\_req*,  
718 *NWK\_addr\_req* and *Match\_Desc\_req* and respond with the *Active\_EP\_rsp*,

- 719 *Node\_Desc\_rsp*, *Simple\_Desc\_rsp*, *IEEE\_addr\_rsp*, *NWK\_addr\_rsp* and  
720 *Match\_Desc\_rsp* commands, respectively.
- 721 • A node SHALL process the ZDO node manager service commands  
722 *Mgmt\_Bind\_req* and *Mgmt\_Lqi\_req* and respond with the *Mgmt\_Bind\_rsp* and  
723 *Mgmt\_Lqi\_rsp* commands, respectively.
- 724 • A node SHALL process the ZDO binding table service commands *Bind\_req*  
725 and *Unbind\_req* and respond with the *Bind\_rsp* and *Unbind\_rsp* commands,  
726 respectively.
- 727 • A node SHALL process the ZDO network manager service command  
728 *Mgmt\_Leave\_req* and respond with the *Mgmt\_Leave\_rsp* command.
- 729 • A node SHALL be able to handle receiving at least one *Identify* cluster,  
730 *Identify Query Response* command frame after broadcasting an *Identify Query*  
731 command frame during finding & binding. If the node is able to handle  
732 receiving more than one *Identify Query Response* command frames, how this  
733 is handled is implementation specific.
- 734 • A node that supports finding & binding as an initiator SHALL implement a  
735 binding table whose number of available entries is greater than or equal to the  
736 sum of the cluster instances, supported on each device of the node, that are  
737 initiators of application transactions. Bindings are configured in the binding  
738 table during finding & binding, touchlink or centralized commissioning.  
739 Regardless of the commissioning mechanism used to generate the bindings,  
740 the binding table SHALL be consistent such that its contents can be retrieved  
741 using the *Mgmt\_Bind\_req* command.
- 742 • A node SHALL have a default report configuration (see sub-clause 6.7) for  
743 every implemented attribute that is specified as mandatory and reportable.
- 744 • A node that can be a target of an application transaction SHALL support  
745 group addressing and at least 8 memberships in the group table.

## 746 **6.7 Default reporting configuration**

747 A default report configuration (with a maximum reporting interval either of 0x0000 or  
748 in the range 0x003d to 0xfffe) SHALL exist for every implemented attribute that is  
749 specified as reportable. The default reporting configuration is such that if a binding is  
750 created on the node to a given cluster the node SHALL send reports to that binding  
751 without any additional reporting configuration needing to be set. The default  
752 reporting configuration for an attribute MAY be overwritten at any time. In this case,  
753 the updated reporting configuration SHALL be used.

754 A report SHALL be generated when the time that has elapsed since the previous  
755 report of the same attribute is equal to the Maximum Reporting Interval for that  
756 attribute. The time of the first report after configuration is not specified. If the  
757 Maximum Reporting Interval is set to 0x0000, there is no periodic reporting, but  
758 change based reporting is still operational.

759 As an example of a default reporting configuration consider a simple humidity sensor.  
760 The humidity sensor knows best what its reporting configuration should be in order to

761 conserve battery power. It should therefore have a default reporting configuration so  
762 that once it is joined to a network, and a binding is created, it would immediately  
763 begin sending reports of its humidity.

## 764 **6.8 MAC data polling**

765 MAC Data polling is required by all sleepy ZigBee end devices to operate correctly  
766 in a ZigBee-PRO network. The Base Device Behavior Specification puts no  
767 restrictions on the frequency of MAC data polls. The choice of how frequently data  
768 polling is done will be based on individual product design considerations to reduce  
769 power consumption. However the following are a set of recommendations to ensure  
770 correct operation in the network:

771 The MAC data polling rate **SHOULD** be dynamic based on the operating state of the  
772 node. It is **RECOMMENDED** it has at least two rates, a fast rate and a slow rate.

773 The ZigBee specification only requires that parent nodes buffer a single message for  
774 7.5 seconds. This single buffer applies to all sleepy ZigBee end devices. Therefore a  
775 sleepy ZigBee end device **SHOULD** poll more frequently than once per 7.5 seconds in  
776 order to be able to retrieve a buffered message that it is expecting.

777 When the node is waiting for an active response message such as an APS  
778 acknowledgement, or a ZCL response, or participating in a multi-message protocol, it  
779 **SHOULD** poll at its fast rate. This fast rate is **RECOMMENDED** to be at least once  
780 every 3 seconds.

781 When the node is not actively waiting for messages it **MAY** poll at its slow rate, for  
782 example, once per hour. This ensures it still has a connection with the network and  
783 with its parent.

784 During initial joining to the ZigBee-PRO network, including finding & binding, the  
785 sleepy ZigBee end device **SHOULD** poll at its fast rate.

## 786 **6.9 ZigBee persistent data**

787 In addition to the persistent data specified in the ZigBee specification (see [R1]) and  
788 the ZCL specification (see [R2]), a node **SHALL** preserve the following data across  
789 resets:

- 790 • *bdbNodeIsOnANetwork* attribute.

791



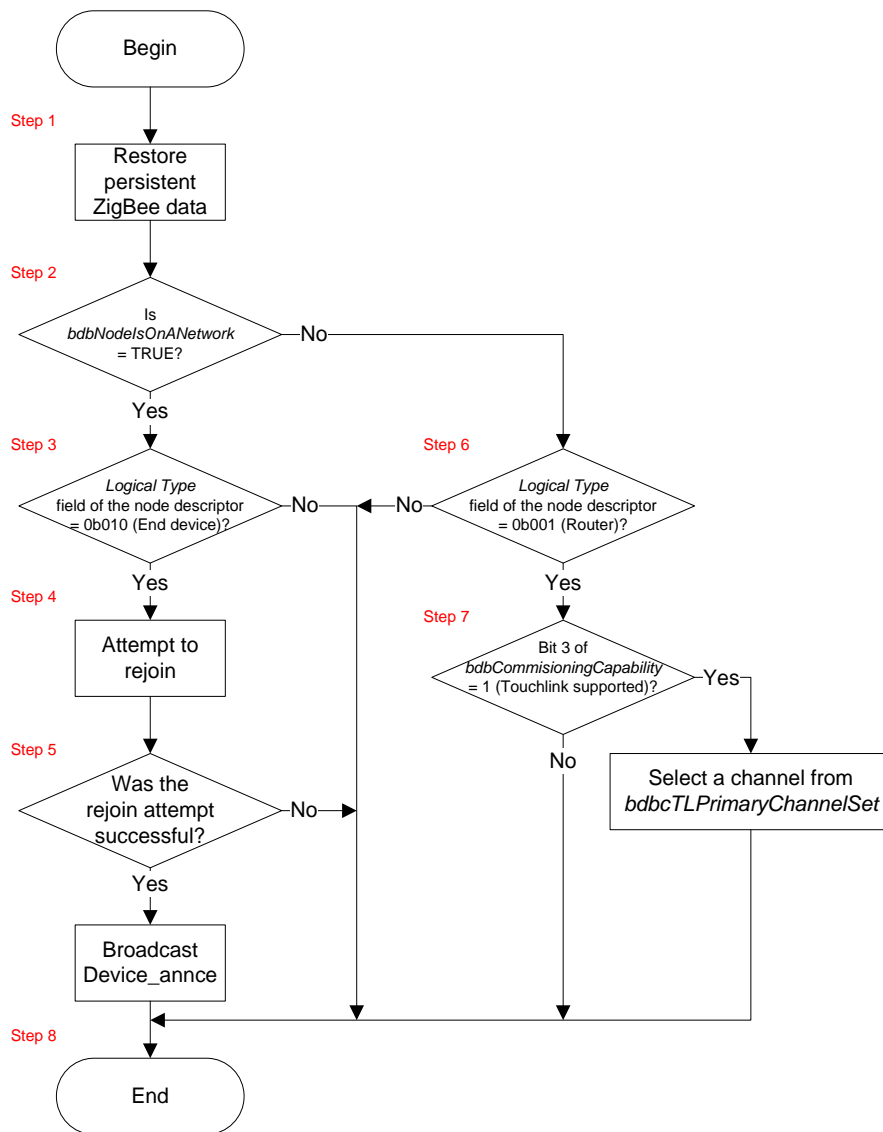
## 792 **7 Initialization**

793 A node performs initialization whenever it is supplied with power either the first time  
 794 or subsequent times after some form of power outage or power-cycle. The ZigBee  
 795 specification (see [R1]) and sub-clause 6.9 defines what data a node is expected to  
 796 preserve through resets and this is restored first to determine how to initialize the  
 797 node. If the node is a router, it is RECOMMENDED that an attempt is first made to  
 798 discover whether its network still exists or has moved to another channel and to take  
 799 corrective action accordingly.

### 800 **7.1 Initialization procedure**

801 This section defines the initialization procedure for a node. Figure 1 illustrates a  
 802 simplified version of this procedure for quick reference.

803



804

805

**Figure 1 – Initialization procedure**

806

- 807 1. The node SHALL restore its persistent ZigBee data, as specified in sub-clause  
808 6.9.
- 809 2. If *bdbNodeIsOnANetwork* is equal to FALSE, the node SHALL continue from  
810 step 6.
- 811 3. If the *logical type* field of the node descriptor for the node is not equal to  
812 0b010 (ZigBee end device), it SHALL continue from step 8.
- 813 4. The node SHALL attempt to rejoin the network. To do this, the node issues  
814 the *NLME-JOIN.request* primitive with the *ExtendedPANId* parameter set to  
815 the extended PAN identifier of the known network, the *RejoinNetwork*  
816 parameter set to 0x02, the *ScanChannels* parameter set to 0x00000000, the  
817 *ScanDuration* parameter set to 0x00, the *CapabilityInformation* set  
818 appropriately for the node and the *SecurityEnable* parameter set to TRUE. On  
819 receipt of the *NLME-JOIN.confirm* primitive from the NWK layer, the node is  
820 notified of the status of the request to join the network using NWK rejoin.
- 821 5. If the *Status* parameter of the *NLME-JOIN.confirm* primitive is equal to  
822 *SUCCESS*, the node SHALL broadcast a *Device\_annce* ZDO command and  
823 continue from step 8. If the *Status* parameter of the *NLME-JOIN.confirm*  
824 primitive is not equal to *SUCCESS*, the node MAY retry the procedure at  
825 some application specific time or continue from step 8. It is the responsibility  
826 of the implementation to handle the subsequent rejoin attempt.
- 827 6. If the *logical type* field of the node descriptor for the node is not equal to  
828 0b001 (ZigBee router), it SHALL continue from step 8.
- 829 7. If bit 3 of *bdbNodeCommissioningCapability* is equal to 1 (touchlink  
830 supported), the node SHALL set its logical channel to one of those specified in  
831 *bdbcTLPrimaryChannelSet*.
- 832 8. The node SHALL then terminate the initialization procedure.
- 833



## 834 **8 Commissioning**

835 Commissioning MAY be invoked when a node is not on a network, on a network but  
836 not bound to another device or on a network and bound to another device.

837 Commissioning MAY be triggered by a user interaction, via some over the air  
838 mechanism (such as that defined in the *Identify* cluster) or invoked directly by  
839 application software (such as automatically after initialization). The commissioning  
840 procedures specified in this section define the steps and states when commissioning is  
841 invoked.

842 An implementation SHALL provide a mechanism to invoke commissioning with  
843 network steering (see sub-clauses 8.2 and 8.3). In addition, a simple device SHALL  
844 provide a mechanism to invoke commissioning with finding & binding (see sub-  
845 clauses 8.5 and 8.6). Similarly, if finding & binding is supported, a dynamic device  
846 SHALL provide a mechanism to invoke commissioning with finding & binding. If  
847 required by the application these commissioning actions MAY be overloaded. An  
848 implementation MAY also provide separate or overloaded mechanisms for other  
849 commissioning actions.

850 The commissioning procedure is controlled per endpoint via the  
851 *bdbCommissioningMode* attribute and this SHOULD be configured, as appropriate,  
852 on each application stimulus before commissioning commences. This allows, for  
853 example, an implementation to overload an application stimulus with both network  
854 steering and finding & binding.

### 855 **8.1 Top level commissioning procedure**

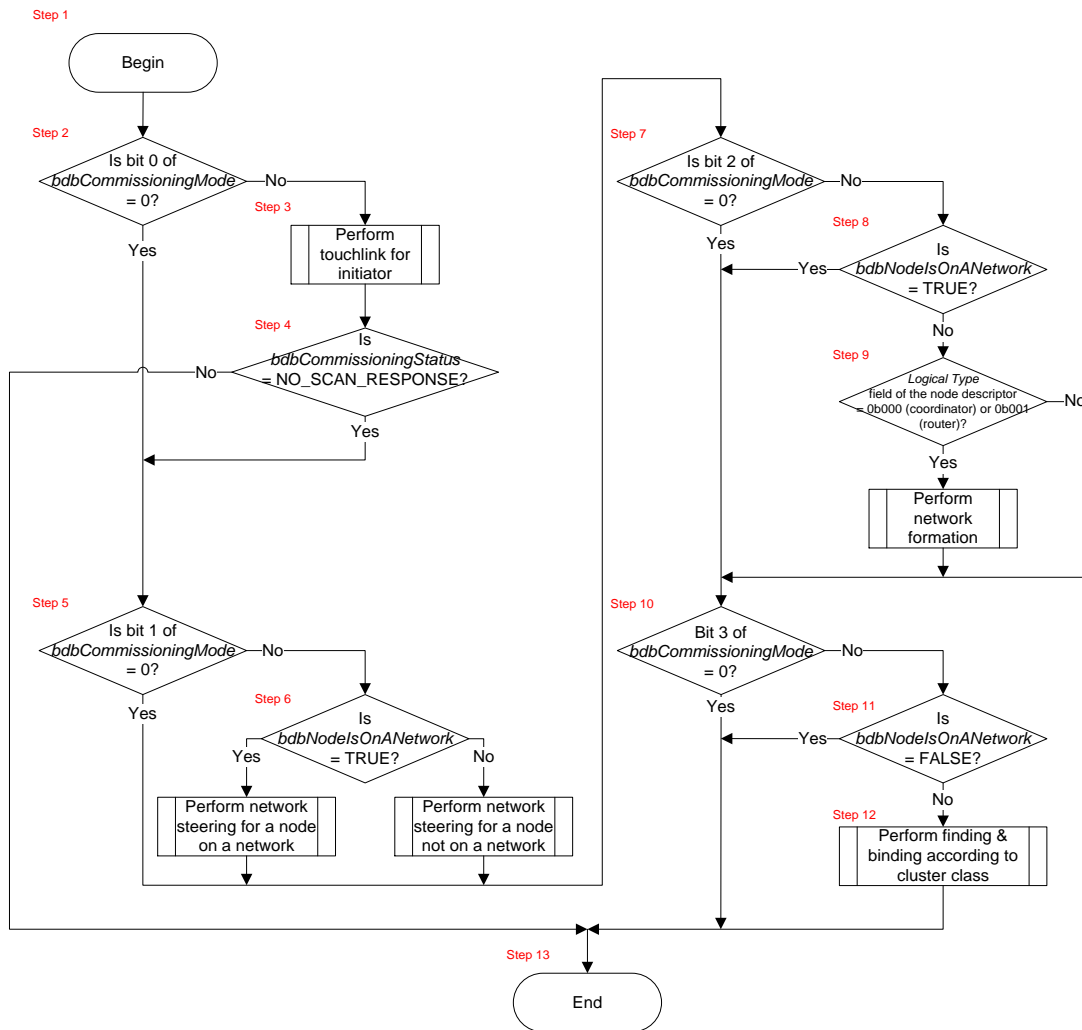
856 This section defines the top level commissioning procedure that is activated on some  
857 trigger.

858 The trigger is via some application defined stimulus, such as a button press or via  
859 some command from a user interface. The stimulus can be per endpoint or on the  
860 node as a whole. The criterion under which this can occur is manufacturer specific.

861 The required commissioning action is configured by the application by setting the  
862 *bdbCommissioningMode* attribute on the desired endpoint to the appropriate values  
863 (see sub-clause 5.3.2) and then following this procedure.

864 Figure 2 illustrates a simplified version of this procedure for quick reference.

865



866  
867

868  
869

**Figure 2 – Top level commissioning procedure**

- 870 1. On receipt of an application stimulus for commissioning, the device first sets
- 871 *bdbCommissioningStatus* to SUCCESS and then determines the required
- 872 commissioning steps by inspecting *bdbCommissioningMode*.
- 873 2. If bit 0 of *bdbCommissioningMode* is equal to 0 (i.e. touchlink is not required),
- 874 the device SHALL continue from step 5.
- 875 3. The node SHALL follow the touchlink procedure as an initiator (see sub-
- 876 clause 8.7).
- 877 4. If *bdbCommissioningStatus* is not equal to NO\_SCAN\_RESPONSE (i.e. there
- 878 was a response to the touchlink scan request from the initiator, indicating a
- 879 successful touchlink), the device SHALL continue from step 13.
- 880 5. If bit 1 of *bdbCommissioningMode* is equal to 0 (i.e. network steering is not
- 881 required), the device SHALL continue from step 7.

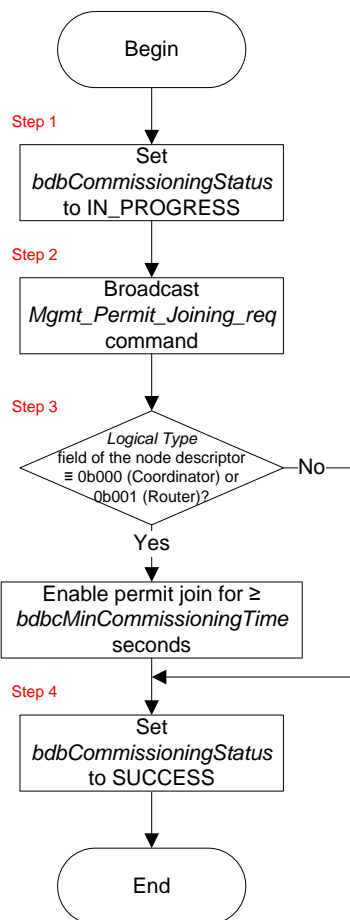
- 882 6. If *bdbNodeIsOnANetwork* is equal to TRUE, the node SHALL follow the  
883 network steering procedure for a node on a network (see sub-clause 8.2). If  
884 *bdbNodeIsOnANetwork* is equal to FALSE, the node SHALL follow the  
885 network steering procedure for a node not on a network (see sub-clause 8.3).  
886 7. If bit 2 of *bdbCommissioningMode* is equal to 0 (i.e. forming a network is not  
887 required), the device SHALL continue from step 10.  
888 8. If *bdbNodeIsOnANetwork* is equal to TRUE, the device SHALL continue  
889 from step 10.  
890 9. If the *logical type* field of the node descriptor for the node is equal to 0b000  
891 (ZigBee coordinator) or 0b001 (ZigBee router), the node SHALL follow the  
892 network formation procedure (see sub-clause 8.4).  
893 10. If bit 3 of *bdbCommissioningMode* is equal to 0 (i.e. finding & binding is not  
894 required), the device SHALL continue from step 13.  
895 11. If *bdbNodeIsOnANetwork* is equal to FALSE, the device SHALL continue  
896 from step 13.  
897 12. If bit 3 of *bdbCommissioningMode* is equal to 1, the node SHALL follow the  
898 finding & binding procedure as appropriate for the class of the clusters  
899 implemented on the endpoints defined on the node. For a type 1 client or a  
900 type 2 server cluster, the application SHALL perform finding & binding as an  
901 initiator endpoint (see sub-clause 8.6). Conversely, for a type 1 server or type  
902 2 client cluster, the application SHALL perform finding & binding as a target  
903 endpoint (see sub-clause 8.5). Note that it is also the responsibility of the  
904 application to determine the order in which the finding & binding is performed  
905 when more than one device endpoints are commissioned and whether some  
906 can be handled in parallel.  
907 13. The device SHALL terminate the top level commissioning procedure.  
908

## 909 **8.2 Network steering procedure for a node on a network**

910 This section defines the network steering procedure for a node that is already on a  
911 network. In this procedure, a node that is already on a network opens up the network  
912 for a finite duration to allow other nodes to join.

913 Figure 3 illustrates a simplified version of this procedure for quick reference.

914



915

916

**Figure 3 – Network steering procedure for a node on a network**

917

- 918 1. The node first sets *bdbCommissioningStatus* to *IN\_PROGRESS*.
- 919 2. The node SHALL broadcast the *Mgmt\_Permit\_Joining\_req* ZDO command  
920 with the *PermitDuration* field set to at least *bdbcMinCommissioningTime* and  
921 the *TC\_Significance* field set to 0x01.
- 922 3. If the *logical type* field of the node descriptor for the node is equal to 0b000  
923 (ZigBee coordinator) or 0b001 (ZigBee router), the node issues the *NLME-*  
924 *PERMIT-JOINING.request* primitive with the *PermitDuration* parameter set  
925 to at least *bdbcMinCommissioningTime*. On receipt of the *NLME-PERMIT-*  
926 *JOINING.confirm* primitive from the NWK layer, the node is notified of the  
927 status of the request to activate permit joining.
- 928 4. The node then sets *bdbCommissioningStatus* to *SUCCESS* and it SHALL  
929 terminate the network steering procedure for a node on a network.

930

### 8.3 Network steering procedure for a node not on a network

931

932

933

934

This section defines the network steering procedure for a node that is not yet on a network. In this procedure, a node that is not already on a network scans for open networks and if a suitable one is found attempts to join. After joining the node is authenticated and receives the network key. Finally, if a Trust Center is present in the

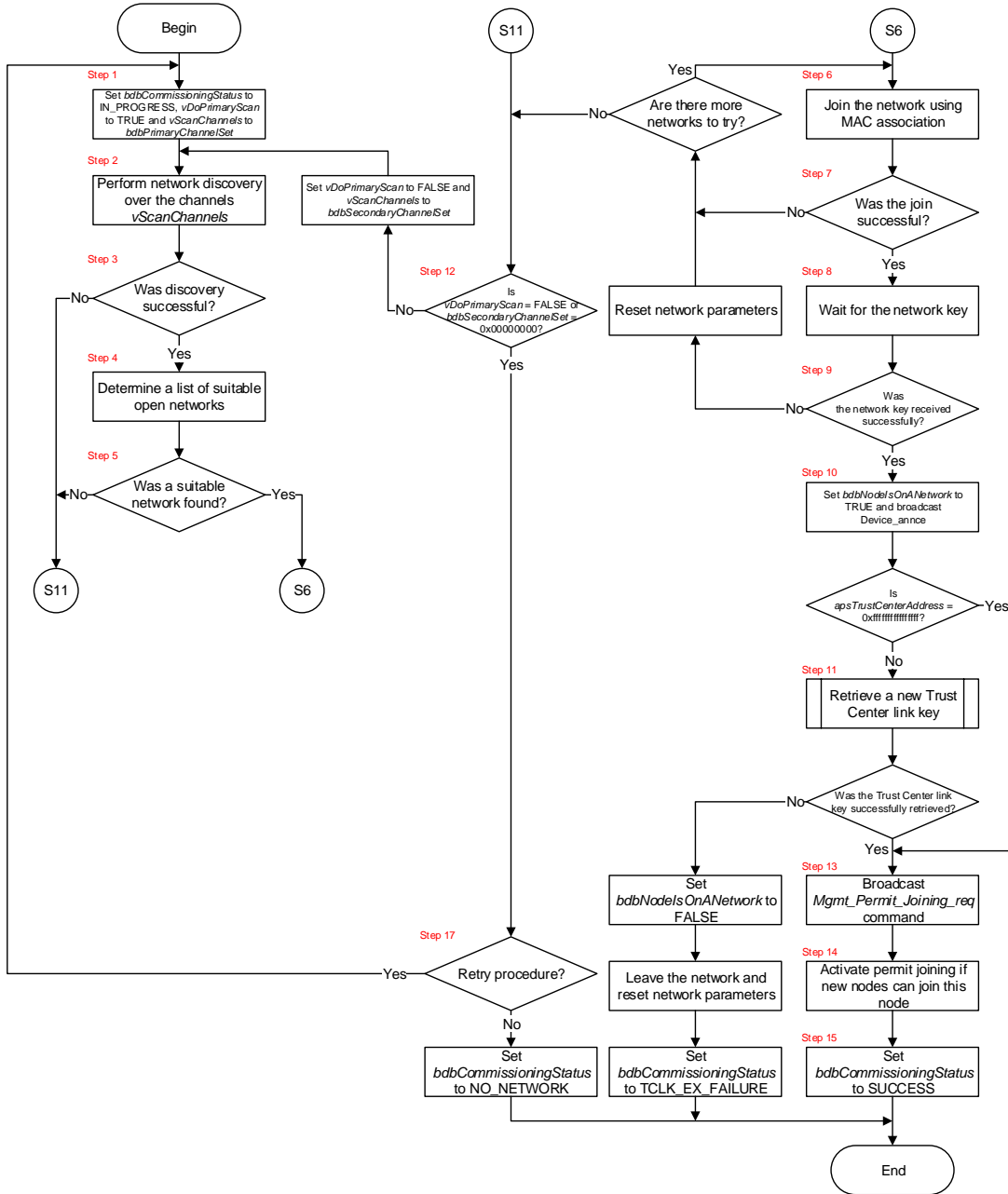
935 network, the node then exchanges its preconfigured link key for one generated by the  
936 Trust Center.

937 Two variables are defined for this procedure: a Boolean value, *vDoPrimaryScan*,  
938 which controls whether a node is to perform a channel scan over the primary or  
939 secondary channel sets and a 32-bit bitmap, *vScanChannels*, which defines the current  
940 set of channels over which to scan.

941 Figure 4 illustrates a simplified version of this procedure for quick reference.

942

943



944  
945

946

947

948

**Figure 4 – Network steering procedure for a node not on a network**

949

950

951

952

953

954

1. The node first sets *bdbCommissioningStatus* to IN\_PROGRESS, *vDoPrimaryScan* to TRUE and *vScanChannels* set to *bdbPrimaryChannelSet*. If *bdbPrimaryChannelSet* is equal to 0x00000000, the node SHALL continue from step 12.
2. The node SHALL perform a channel scan in order to discover which networks are available within its radio range on a set of channels. To do this, the node

- 955 issues the *NLME-NETWORK-DISCOVERY.request* primitive with the  
956 *ScanChannels* parameter set to *vScanChannels* and the *ScanDuration*  
957 parameter set to *bdbScanDuration*. On receipt of the *NLME-NETWORK-*  
958 *DISCOVERY.confirm* primitive from the NWK layer, the node is notified of  
959 the status of the request to discover networks.
- 960 3. If the *Status* parameter from the *NLME-NETWORK-DISCOVERY.confirm*  
961 primitive is not equal to *SUCCESS*, indicating that the channel scan was not  
962 successful, the node SHALL continue from step 12.
  - 963 4. The node SHALL determine whether any suitable networks with a permit  
964 joining flag set to TRUE were found by analyzing the *NetworkCount* and  
965 *NetworkDescriptor* parameters. The decision regarding what constitutes a  
966 *suitable* network is application specific.
  - 967 5. If a suitable network is not found on the channel scan, the node SHALL  
968 continue from step 12.
  - 969 6. The node SHALL attempt to join the network found using MAC association.  
970 To do this, the node issues the *NLME-JOIN.request* primitive with the  
971 *ExtendedPANId* parameter set to the extended PAN identifier of the selected  
972 network, the *RejoinNetwork* parameter set to 0x00, the *ScanChannels*  
973 parameter set to 0x00000000, the *ScanDuration* parameter set to 0x00, the  
974 *CapabilityInformation* set appropriately for the node and the *SecurityEnable*  
975 parameter set to FALSE. On receipt of the *NLME-JOIN.confirm* primitive  
976 from the NWK layer, the node is notified of the status of the request to join the  
977 network using MAC association.
  - 978 7. If the *Status* parameter from the *NLME-JOIN.confirm* primitive is not equal to  
979 *SUCCESS*, indicating that the join was not successful, the node SHALL try to  
980 join the next suitable network from step 6. Note that it is permissible to try to  
981 join the same network again, but this SHALL NOT be attempted more than  
982 *bdbcMaxSameNetworkRetryAttempts* times in succession (*bdbcRecSame-*  
983 *NetworkRetryAttempts* times in succession is RECOMMENDED). If there are  
984 no further suitable networks to join the node SHALL continue from step 12.
  - 985 8. If the *Status* parameter from the *NLME-JOIN.confirm* primitive is equal to  
986 *SUCCESS*, indicating that the join was successful, the node SHALL wait for at  
987 most *apsSecurityTimeOutPeriod* milliseconds to be authenticated and receive  
988 the network key from its parent. Note that the network key may be tunneled  
989 from the Trust Center in a centralized security network, encrypted using the  
990 default global Trust Center link key or via an install code derived  
991 preconfigured link key, or directly from its parent in a distributed security  
992 network, encrypted using the distributed security global link key. The node  
993 SHALL set *bdbNodeJoinLinkKeyType* accordingly to indicate the type of link  
994 key used to decrypt the received network key.
  - 995 9. If the node does not receive the network key from its parent within  
996 *apsSecurityTimeOutPeriod* milliseconds, the network key is received within



997           *apsSecurityTimeOutPeriod* milliseconds but cannot be decrypted or the  
998           authentication fails in some other way, the node SHALL reset its network  
999           parameters and select the next suitable network to join and return to step 6.  
1000          Note that it is permissible to try to join the same network again, but this  
1001          SHALL NOT be attempted more than *bdbcMaxSameNetworkRetryAttempts*  
1002          times in succession (*bdbcRecSameNetworkRetryAttempts* times in succession  
1003          is RECOMMENDED). If there are no further suitable networks to join, the  
1004          node SHALL continue from step 12.

- 1005          10. The node sets *bdbNodeIsOnANetwork* to TRUE and then broadcasts a  
1006             Device\_annce ZDO command. If *apsTrustCenterAddress* is equal to  
1007             0xffffffffffffff, the node SHALL continue from step 13.
- 1008          11. The node SHALL perform the procedure for retrieving a new Trust Center  
1009             link key (see sub-clause 10.2.5). If the procedure is successful, the node  
1010             SHALL continue from step 13. If the procedure is not successful, the node  
1011             SHALL perform a leave request on its old network and resets its network  
1012             parameters. The node then sets *bdbNodeIsOnANetwork* to FALSE and sets  
1013             *bdbCommissioningStatus* to *TCLK\_EX\_FAILURE*. To perform a leave  
1014             request, the node issues the *NLME-LEAVE.request* primitive to the NWK  
1015             layer with the *DeviceAddress* parameter set to NULL, the *RemoveChildren*  
1016             parameter set to FALSE and the *Rejoin* parameter set to FALSE. On receipt  
1017             of the *NLME-LEAVE.confirm* primitive, the node is notified of the status of  
1018             the request to leave the network. The node SHALL then terminate the  
1019             network steering procedure for a node not on a network.
- 1020          12. If *vDoPrimaryScan* is equal to FALSE or *bdbSecondaryChannelSet* is equal to  
1021             0x00000000, the node SHALL continue from step 16. If  
1022             *bdbSecondaryChannelSet* is not equal to 0x00000000, the node SHALL set  
1023             *vDoPrimaryScan* to FALSE, set *vScanChannels* to *bdbSecondaryChannelSet*  
1024             and continue from step 2.
- 1025          13. The node SHALL broadcast the *Mgmt\_Permit\_Joining\_req* ZDO command  
1026             with the *PermitDuration* field set to at least *bdbcMinCommissioningTime* and  
1027             the *TC\_Significance* field set to 0x01. Note that this will cause nodes  
1028             receiving this command to reset the timer, during which their permit joining  
1029             flag is activated, thus extending the time for further new nodes to join.
- 1030          14. If the node is able to allow new nodes to join, it SHALL activate its permit  
1031             joining flag. To do this, the node issues the *NLME-PERMIT-*  
1032             *JOINING.request* primitive with the *PermitDuration* parameter set to at least  
1033             *bdbcMinCommissioningTime*. On receipt of the *NLME-PERMIT-*  
1034             *JOINING.confirm* primitive from the NWK layer, the node is notified of the  
1035             status of the request to activate permit joining.
- 1036          15. The node then sets *bdbCommissioningStatus* to SUCCESS. If the node  
1037             supports touchlink, it sets the values of the *aplFreeNwkAddrRangeBegin*,  
1038             *aplFreeNwkAddrRangeEnd*, *aplFreeGroupID-RangeBegin* and



1039           *aplFreeGroupIDRangeEnd* attributes all to 0x0000 (indicating the node  
1040           having joined the network using MAC association). The node SHALL then  
1041           terminate the network steering procedure for a node not on a network.  
1042       16. The node MAY retry using some manufacturer specific procedure OR set  
1043           *bdbCommissioningStatus* to NO\_NETWORK and then it SHALL terminate  
1044           the network steering procedure for a node not on a network. If a manufacturer  
1045           specific procedure is attempted, the *bdbCommissioningStatus* and  
1046           *bdbNodeIsOnANetwork* attributes are updated accordingly on its termination  
1047           so that the commissioning procedure is consistent.

#### 1048   **8.4 Network formation procedure**

1049   This section defines the network formation procedure for a node. In this procedure, a  
1050   ZigBee coordinator node forms a centralized security network and activates its Trust  
1051   Center functionality whereas a ZigBee router node forms a distributed security  
1052   network.

1053   Two variables are defined for this procedure: a Boolean value, *vDoPrimaryScan*,  
1054   which controls whether a node is to perform a channel scan over the primary or  
1055   secondary channel sets and a 32-bit bitmap, *vScanChannels*, which defines the current  
1056   set of channels over which to scan.

1057   Figure 5 illustrates a simplified version of this procedure for quick reference.

1058

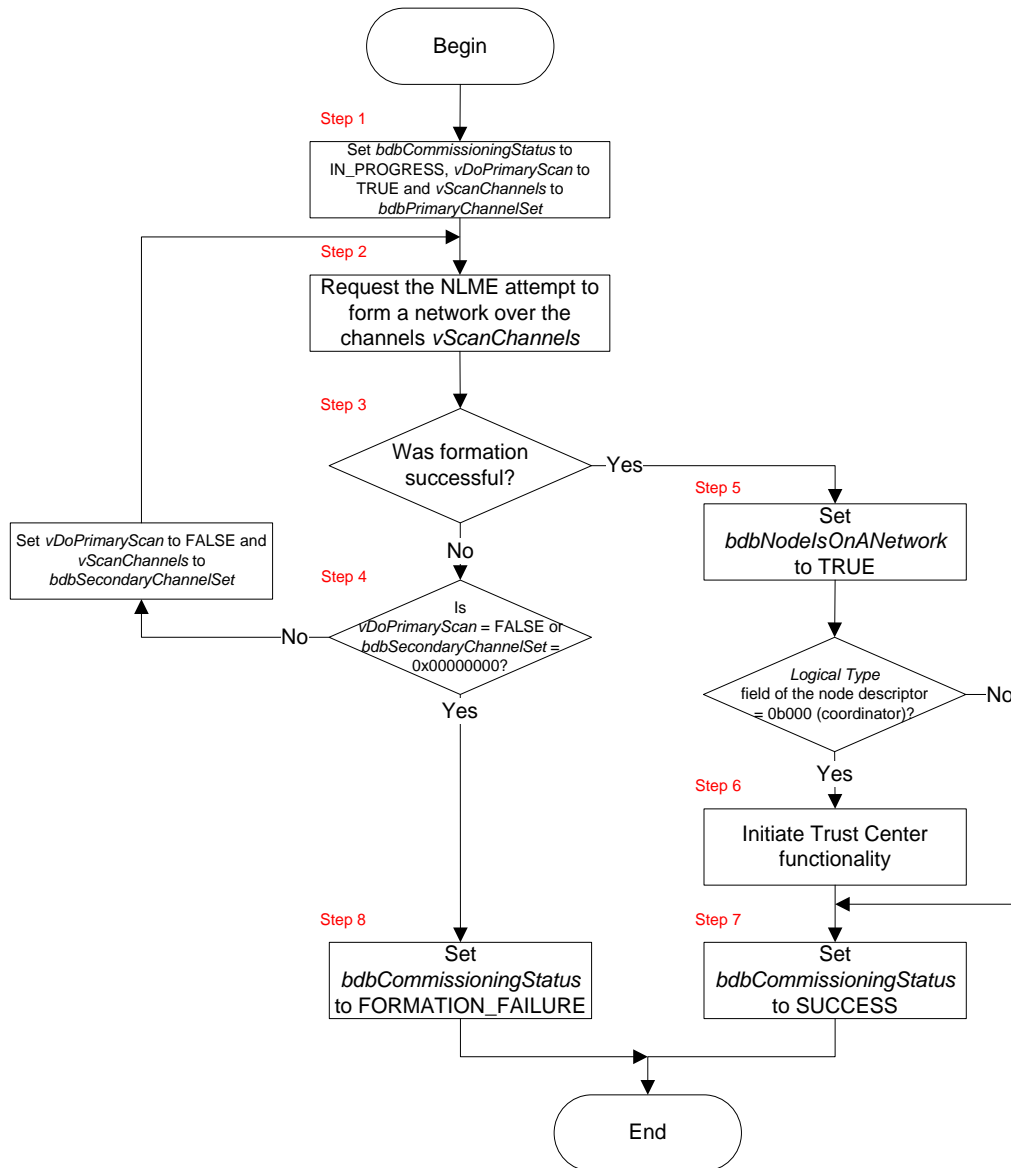


Figure 5 – Network formation procedure

1059  
1060  
1061  
1062  
1063  
1064  
1065  
1066  
1067  
1068  
1069  
1070  
1071  
1072  
1073

1. The node first sets *bdbCommissioningStatus* to *IN\_PROGRESS*, *vDoPrimaryScan* to *TRUE* and *vScanChannels* to *bdbPrimaryChannelSet*. If *bdbPrimaryChannelSet* is equal to *0x00000000*, the node SHALL continue from step 4.
2. The node SHALL attempt to form a network on one of the specified channels. To do this, the node issues the *NLME-NETWORK-FORMATION.request* primitive with the *ScanChannels* parameter set to *vScanChannels*, the *ScanDuration* parameter set to *bdbScanDuration*, the *BeaconOrder* parameter set to *0x0f*, the *SuperframeOrder* set to *0x00* and the *BatteryLifeExtension* parameter set to *FALSE*. On receipt of the *NLME-NETWORK-FORMATION.confirm* primitive from the NWK layer, the node is notified of the status of the request to form a new network.

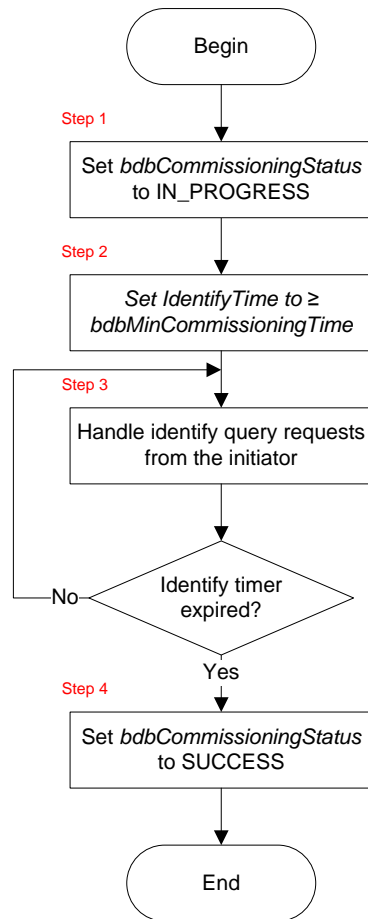
- 1074 3. If the *Status* parameter of the *NLME-NETWORK-FORMATION.confirm*  
1075 primitive is equal to SUCCESS, indicating that a new network has been  
1076 formed, the node SHALL continue from step 5.
- 1077 4. If *vDoPrimaryScan* is equal to FALSE or *bdbSecondaryChannelSet* is equal to  
1078 0x00000000, the node SHALL continue from step 8. If  
1079 *bdbSecondaryChannelSet* is not equal to 0x00000000, the node SHALL set  
1080 *vDoPrimaryScan* to FALSE, set *vScanChannels* to *bdbSecondaryChannelSet*  
1081 and continue from step 2.
- 1082 5. The node sets *bdbNodeIsOnANetwork* to TRUE. If the *logical type* field of  
1083 the node descriptor for the node is not equal to 0b000 (ZigBee coordinator),  
1084 the node SHALL continue from step 7.
- 1085 6. The ZigBee coordinator node SHALL then initiate its Trust Center  
1086 functionality according to sub-clause 4.6.1 of [R1].
- 1087 7. The node then sets *bdbCommissioningStatus* to SUCCESS and it SHALL  
1088 terminate the network formation procedure.
- 1089 8. The node sets *bdbCommissioningStatus* to FORMATION\_FAILURE and it  
1090 SHALL terminate the network formation procedure.

1091

## 1092 **8.5 Finding & binding procedure for a target endpoint**

1093 This section defines the finding & binding procedure for a target endpoint. In this  
1094 procedure, the target endpoint identifies itself for a finite duration and then handles  
1095 subsequent finding & binding requests from an initiator endpoint.

1096 Figure 6 illustrates a simplified version of this procedure for quick reference.



1097

1098

**Figure 6 – Finding & binding procedure for a target endpoint**

1099

- 1100 1. The target device first sets *bdbCommissioningStatus* to IN\_PROGRESS.
- 1101 2. The target device SHALL set the *Identify* cluster, *IdentifyTime* attribute to at
- 1102 at least *bdbMinCommissioningTime*. The target device MAY also set the
- 1103 *Identify* cluster, *IdentifyTime* attribute to at least *bdbMinCommissioningTime*
- 1104 on any other identifying endpoints.
- 1105 3. During the *IdentifyTime*, the target device SHALL respond to the identify
- 1106 queries. They may be followed by other finding & binding commands; those
- 1107 SHALL be handled irrespective of the identify status.
- 1108 4. When the decrementing *IdentifyTime* attribute reaches zero, the target device
- 1109 sets *bdbCommissioningStatus* to SUCCESS and it SHALL terminate the
- 1110 finding & binding procedure for a target endpoint.

1111

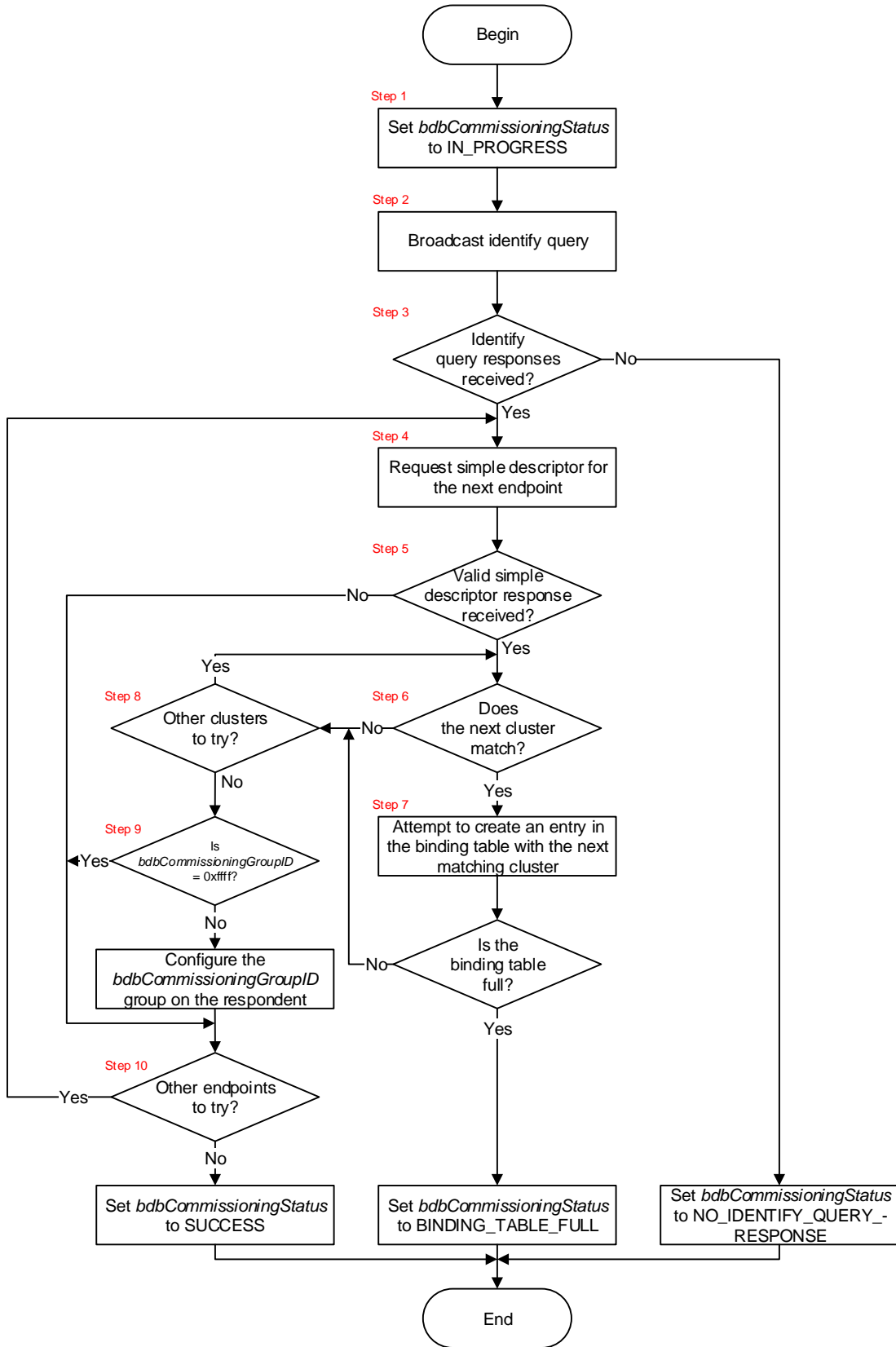
## 8.6 Finding & binding procedure for an initiator endpoint

1112

1113 This section defines the finding & binding procedure for an initiator endpoint. In this  
 1114 procedure, the initiator endpoint first searches for identifying target endpoints and if  
 1115 one is found, its simple descriptor is requested. The initiator endpoint then searches  
 for any matching clusters between itself and the target endpoint and for each match

- 1116 found, it creates a corresponding entry in its binding table. If a group binding is  
1117 requested, the initiator endpoint configures group membership of the target endpoint.  
1118 Figure 7 illustrates a simplified version of this procedure for quick reference.

1119



1120  
1121

1122

**Figure 7 – Finding & binding procedure for an initiator endpoint**

1123

- 1124 1. The initiator device first sets *bdbCommissioningStatus* to IN\_PROGRESS.
- 1125 2. The initiator device SHALL broadcast the *Identify* cluster, *Identify Query*
- 1126 *command* from the initiator endpoint to all nodes (i.e., using the broadcast
- 1127 *address 0xffff*). The initiator device MAY broadcast this command one or
- 1128 more times.
- 1129 3. If no *Identify* cluster, *Identify Query Response* commands are received, the
- 1130 initiator device sets *bdbCommissioningStatus* to
- 1131 NO\_IDENTIFY\_QUERY\_RESPONSE and it SHALL terminate the finding
- 1132 & binding procedure for an initiator endpoint. If at least one *Identify* cluster,
- 1133 *Identify Query Response* command is received, the initiator device SHALL
- 1134 note the NWK address, contained in the *source address* field of the NWK
- 1135 header, and the endpoint, contained in the *source endpoint* field of the APS
- 1136 header, of each incoming frame from the target devices that responded; such a
- 1137 device is referred to as a “respondent”.
- 1138 4. The initiator device SHALL obtain the simple descriptor for the next response
- 1139 endpoint from a respondent. To do this, the initiator device SHALL unicast
- 1140 the *Simple\_Desc\_req* ZDO command to the respondent with the
- 1141 *NWKAddrOfInterest* field set to the NWK address of the respondent and the
- 1142 *EndPoint* field set to the identifier of the endpoint being addressed (found
- 1143 from the APS header of the respondent’s *Identify* cluster, *Identify Query*
- 1144 *Response* command).
- 1145 5. If a *Simple\_Desc\_rsp* ZDO command is not received from the respondent or a
- 1146 *Simple\_Desc\_rsp* ZDO command is received with the *Status* field not equal to
- 1147 SUCCESS, the initiator device SHALL continue from step 10.
- 1148 6. The initiator SHALL check the next application target cluster listed in the
- 1149 *Application Input Cluster List* or *Application Output Cluster List* fields of the
- 1150 simple descriptor of the respondent and if the initiator device does not support
- 1151 the corresponding client/server cluster, the initiator device SHALL continue
- 1152 from step 8.
- 1153 7. If the initiator is a simple device, it SHALL create a binding table entry for
- 1154 that cluster. Conversely, if the initiator is not a simple device, it MAY create a
- 1155 binding table entry for that cluster. If a unicast binding table entry is to be
- 1156 created (i.e., if *bdbCommissioningGroupId* is equal to 0xffff) and the IEEE
- 1157 address of the respondent is not known, the initiator SHALL obtain it using
- 1158 the *IEEE\_addr\_req* ZDO command before creating a binding. To create a
- 1159 binding table entry, the initiator device issues the *APSME-BIND.request*
- 1160 primitive with the *SrcAddr* parameter set to the IEEE address of the initiator
- 1161 device (*aExtendedAddress*), the *SrcEndpoint* parameter set to the identifier of
- 1162 the initiator endpoint and the *ClusterId* parameter set to the identifier of the
- 1163 matching cluster. The *DstAddrMode* and *DstAddr* parameters SHALL be set
- 1164 to 0x01 and *bdbCommissioningGroupId*, respectively, (if

1165 *bdbCommissioningGroupId* is not equal to 0xffff) or 0x03 and the known  
1166 IEEE address of the respondent, respectively, (if *bdbCommissioningGroupId*  
1167 is equal to 0xffff). The *DstEndpoint* parameter SHOULD be included and set  
1168 to the identifier of the endpoint on the respondent on which the matching  
1169 cluster was found only if *bdbCommissioningGroupId* is equal to 0xffff. On  
1170 receipt of the *APSME-BIND.confirm* primitive from the APS sub-layer, the  
1171 initiator device is notified of the status of the request to create a binding table  
1172 entry. If the *Status* parameter of the *APSME-BIND.confirm* primitive is equal  
1173 to *TABLE\_FULL*, the device sets *bdbCommissioningStatus* to  
1174 *BINDING\_TABLE\_FULL* and it SHALL terminate the finding & binding  
1175 procedure for an initiator endpoint.

- 1176 8. If there are further matching clusters discovered from the simple descriptor,  
1177 the initiator device SHALL select the next one and continue from step 6.
- 1178 9. If *bdbCommissioningGroupID* is not equal to 0xffff and at least one binding  
1179 link was created, the initiator device SHALL either unicast the *groups* cluster,  
1180 *add group* command to the respondent or broadcast the *groups* cluster, *add*  
1181 *group if identifying* command with the *Group ID* field set to  
1182 *bdbCommissioningGroupID*.
- 1183 10. If there are further endpoints discovered via the *Identify Query* command, the  
1184 initiator device SHALL select the next endpoint and continue from step 4. If  
1185 there are no further endpoints to select, the initiator device sets  
1186 *bdbCommissioningStatus* to *SUCCESS* and it SHALL terminate the finding &  
1187 binding procedure for an initiator endpoint. Note: if required by the  
1188 application, the initiator MAY send the *Identify* cluster, *Identify* command  
1189 with the *IdentifyTime* field set to 0x0000 (stop the identify procedure) to all  
1190 the identifying targets.

## 1191 **8.7 Touchlink procedure for an initiator**

1192 This section defines the touchlink procedure for an initiator. In this procedure, the  
1193 node that initiates the touchlink operation is called the “initiator” and the node that  
1194 responds is called the “target”. The initiator scans for nodes also supporting touchlink  
1195 and if one is found establishes a new network with the target (if the initiator is not on  
1196 a network) or adds the target to the network (if the initiator is already on a network).

1197 Three variables are defined for this procedure: a Boolean value, *vDoPrimaryScan*,  
1198 which controls whether a node is to perform a channel scan over the primary or  
1199 secondary channel sets, a 32-bit bitmap, *vScanChannels*, which defines the current set  
1200 of channels over which to scan and a Boolean value, *vIsFirstChannel* which controls  
1201 whether to use the first channel to perform the first five touchlink commissioning  
1202 scans.

1203 The touchlink procedure for an initiator can perform a “normal” channel scan or an  
1204 “extended” channel scan; the latter is used if a reset to factory new is required (see  
1205 sub-clause 9.2) or if the target could be operating on a channel other than those  
1206 defined in *bdbTLPrimaryChannelSet*. For a normal channel scan,

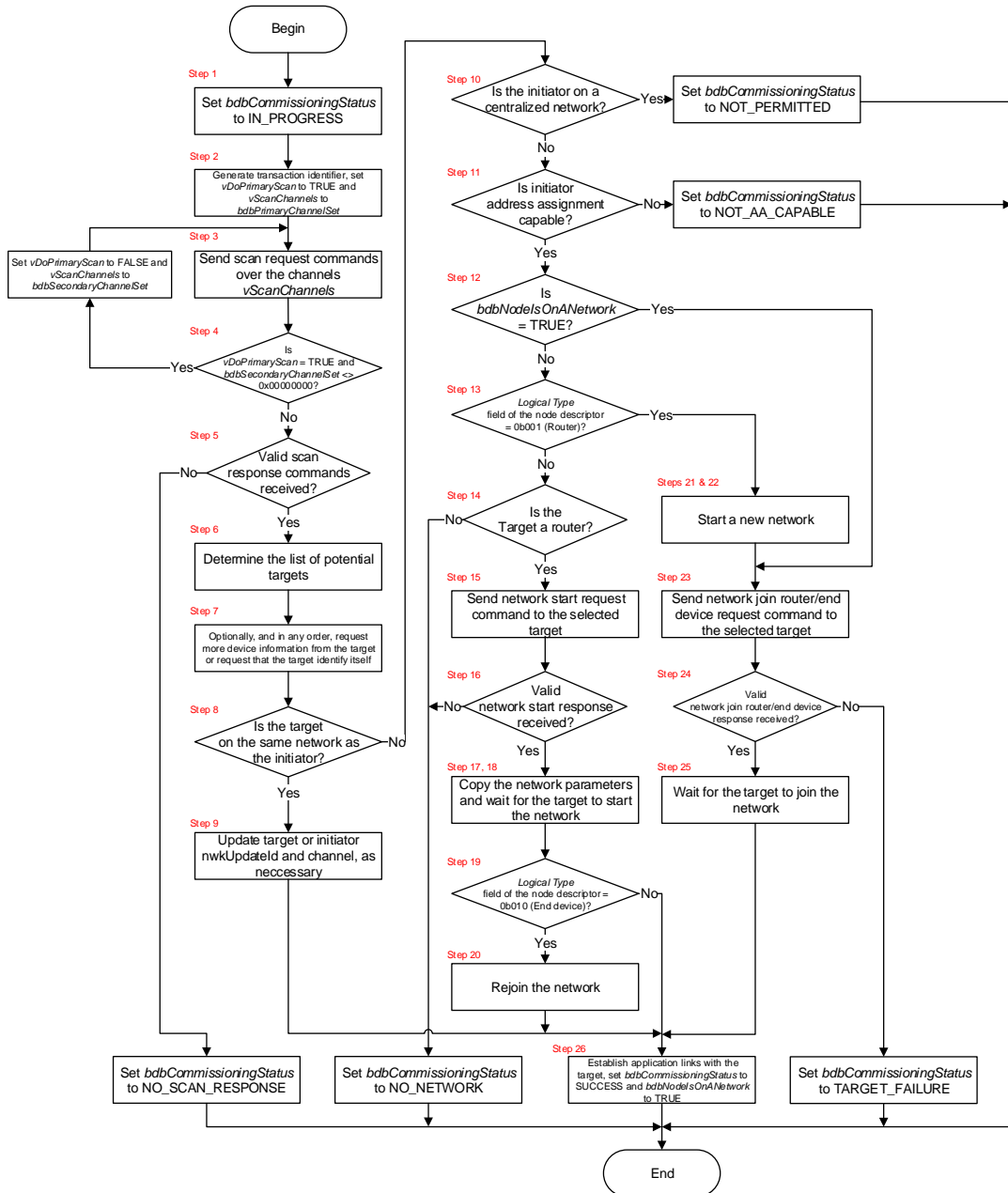


1207 *bdbPrimaryChannelSet* and *bdbSecondaryChannelSet* SHALL be set to  
 1208 *bdbcTLPrimaryChannelSet* and 0x00000000, respectively. For an extended channel  
 1209 scan, *bdbPrimaryChannelSet* and *bdbSecondaryChannelSet* SHALL be set to  
 1210 *bdbcTLPrimaryChannelSet* and *bdbcTLSecondaryChannelSet*, respectively.

1211 Figure 8 illustrates a simplified version of this procedure for quick reference.

1212

1213



1214

1215

1216

1217

**Figure 8 – Touchlink procedure for an initiator**

1. The initiator first sets *bdbCommissioningStatus* to IN\_PROGRESS.

- 1218           2. The initiator SHALL generate a 32-bit transaction identifier to use in the  
1219           *inter-PAN transaction identifier* fields of all commands used in the  
1220           touchlink procedure. The transaction identifier SHALL be random, non-  
1221           zero and non-sequential. The initiator then sets *vDoPrimaryScan* to  
1222           TRUE, *vScanChannels* set to *bdbPrimaryChannelSet* and *vIsFirstChannel*  
1223           set to TRUE. If *bdbPrimaryChannelSet* is equal to 0x00000000, the node  
1224           SHALL continue from step 4.
- 1225           3. The initiator SHALL perform touchlink device discovery. If  
1226           *vIsFirstChannel* is equal to TRUE, the initiator SHALL set  
1227           *vIsFirstChannel* to FALSE, switch to the first channel defined by  
1228           *vScanChannels* and broadcast five consecutive *touchlink commissioning*  
1229           cluster *scan request* inter-PAN command frames. The initiator SHALL  
1230           then switch to each of the remaining channels specified in *vScanChannels*  
1231           in turn and broadcast a single *scan request* inter-PAN command frame on  
1232           each channel. Each *scan request* inter-PAN command frames SHALL be  
1233           broadcast with appropriate values for the *ZigBee information* and *touchlink*  
1234           *information* fields and with a nominal output power of 0dBm. After each  
1235           transmission, the initiator SHALL wait *bdbTLScanTimeBaseDuration*  
1236           seconds to receive any responses. If, during its scan, an initiator with the  
1237           *bdbNodeIsOnANetwork* attribute equal to FALSE receives another *scan*  
1238           *request* inter-PAN command frame with the *factory new* sub-field of the  
1239           *touchlink information* field equal to 1, it SHALL be ignored. Conversely,  
1240           if, during its scan, an initiator with the *bdbNodeIsOnANetwork* attribute  
1241           equal to FALSE receives another *scan request* inter-PAN command frame  
1242           with the *factory new* sub-field of the *touchlink information* field equal to 0,  
1243           it MAY stop sending its own *scan request* inter-PAN command frames and  
1244           assume the role of a target (see sub-clause 8.8), responding with a  
1245           *touchlink commissioning cluster scan response* inter-PAN command frame  
1246           and remaining on the same channel for further touchlink command frames.  
1247           Touchlink device discovery MAY be aborted at any time. Since no node  
1248           parameters such as network settings are altered, this step is non-intrusive  
1249           for the nodes involved.
- 1250           4. If *vDoPrimaryScan* is equal to TRUE and *bdbSecondaryChannelSet* is not  
1251           equal to 0x00000000, the node sets *vDoPrimaryScan* to FALSE, set  
1252           *vScanChannels* to *bdbSecondaryChannelSet* and it SHALL continue from  
1253           step 3.
- 1254           5. If no *touchlink commissioning cluster scan response* inter-PAN command  
1255           frames are received or no *touchlink commissioning cluster scan response*  
1256           inter-PAN command frames are received with the *inter-PAN transaction*  
1257           *identifier* field equal to that used by the initiator in its *scan request*  
1258           command frame, the node sets *bdbCommissioningStatus* to

- 1259 NO\_SCAN\_RESPONSE and it SHALL terminate the touchlink procedure  
1260 for an initiator.
- 1261 6. Touchlink device discovery can result in more than one *touchlink*  
1262 *commissioning cluster scan response* inter-PAN command frames giving a  
1263 list of potential targets from which the application, via some product  
1264 specific means, selects one target for further processing. If the *touchlink*  
1265 *priority request* bit of the *touchlink information* field of the *touchlink*  
1266 *commissioning cluster scan response* command frame is equal to 1, the  
1267 initiator MAY consider giving priority processing to those nodes.
- 1268 7. In any order, the initiator MAY request more device information from the  
1269 target, if necessary, or request the selected target to identify itself in order  
1270 to support a user confirmation. To request more device information from  
1271 the target, the initiator SHALL generate and transmit a *touchlink*  
1272 *commissioning cluster device information request* inter-PAN command  
1273 frame to the appropriate discovered target and wait for a corresponding  
1274 *touchlink commissioning cluster device information response* inter-PAN  
1275 command frame (note that this is not necessary if a target has only one  
1276 sub-device since its information is entirely contained in the *scan response*  
1277 command frame). To request the target identify itself, the initiator SHALL  
1278 generate and transmit a *touchlink commissioning cluster identify request*  
1279 inter-PAN command frame to the appropriate discovered target. The  
1280 initiator MAY send further *identify request* inter-PAN command frames to  
1281 the selected target, for example, to stop the identify operation, provided it  
1282 can do so within *bdbcTLInterPANTransIdLifetime* seconds of the start of  
1283 the touchlink transaction. If this is not possible, a new touchlink device  
1284 discovery operation SHALL be performed.
- 1285 8. If the *extended PAN identifier* field of the *scan response* command frame  
1286 is not equal to *nwkExtendedPANID* (i.e., the target is not on the same  
1287 network as the initiator), the initiator SHALL continue from step 10.
- 1288 9. If the *network update identifier* field of the *scan response* command frame  
1289 is lower than *nwkUpdateId* (i.e., the target has missed a channel change),  
1290 the initiator SHALL generate and transmit a *touchlink commissioning*  
1291 *cluster network update request* command frame to the target with the  
1292 *network update identifier* field set to *nwkUpdateId* and the *logical channel*  
1293 field set to the current operating channel of the initiator. If the *network*  
1294 *update identifier* field of the *scan response* command frame is higher than  
1295 *nwkUpdateId* (i.e., the initiator has missed a channel change), the initiator  
1296 SHALL set *nwkUpdateId* and its current operating channel to the values of  
1297 the *network update identifier* and *logical channel* fields, respectively, from  
1298 the *scan response* command frame. The initiator SHALL continue from  
1299 step 26.

- 1300 10. If the value of *apsTrustCenterAddress* is not equal to 0xffffffffffffff (i.e.,  
1301 the initiator is on a centralized security network), the initiator sets  
1302 *bdbCommissioningStatus* to NOT\_PERMITTED and it SHALL terminate  
1303 the touchlink procedure for an initiator.
- 1304 11. If the initiator is not touchlink address assignment capable, it sets  
1305 *bdbCommissioningStatus* to NOT\_AA\_CAPABLE and it SHALL  
1306 terminate the touchlink procedure for an initiator.
- 1307 12. If *bdbNodeIsOnANetwork* is equal to TRUE, the initiator SHALL continue  
1308 from step 23.
- 1309 13. If the *logical type* field of the node descriptor for the initiator is equal to  
1310 0b001 (ZigBee router), the initiator SHALL continue from step 21.
- 1311 14. If the selected target is not a ZigBee router, the initiator sets  
1312 *bdbCommissioningStatus* to NO\_NETWORK and it SHALL terminate the  
1313 touchlink procedure for an initiator.
- 1314 15. The initiator SHALL generate and unicast a *touchlink commissioning*  
1315 cluster *network start request* inter-PAN command frame to the selected  
1316 target. The initiator SHALL set the *logical channel* field either to zero  
1317 (indicating that the target should choose the channel) or to a channel from  
1318 *bdbcTLPrimaryChannelSet* if a specific primary channel is preferred. The  
1319 initiator SHALL set both the *extended PAN identifier* and *PAN identifier*  
1320 fields to zero. The initiator SHALL also set the *initiator IEEE address* and  
1321 *initiator network address* fields to its IEEE address and the network  
1322 address it will use on the new network, respectively. All other fields  
1323 SHALL be specified according to sub-clause 8.7.1.
- 1324 16. The initiator SHALL then enable its receiver and wait for at most  
1325 *bdbcRxWindowDuration* seconds or until a corresponding *network start*  
1326 *response* inter-PAN command frame is received from the intended target  
1327 with the same *inter-PAN transaction identifier* field matching that used by  
1328 the initiator in its *scan request* command frame. If a corresponding  
1329 *network start response* inter-PAN command frame is not received within  
1330 *bdbcRxWindowDuration* seconds or if a corresponding *network start*  
1331 *response* inter-PAN command frame is received within  
1332 *bdbcRxWindowDuration* seconds but with a non-zero value in the *Status*  
1333 parameter, the initiator sets *bdbCommissioningStatus* to NO\_NETWORK  
1334 and it SHALL terminate the touchlink procedure for an initiator.
- 1335 17. On receipt of a *network start response* inter-PAN command frame with the  
1336 *Status* parameter set to SUCCESS, the initiator SHALL copy these  
1337 parameters to its network information base. The initiator SHALL  
1338 determine whether an entry exists in *apsDeviceKeyPairSet* with a  
1339 *DeviceAddress* field which corresponds to 0xffffffffffffff. If such an  
1340 entry does not exist, the initiator SHALL create a new entry with the  
1341 *DeviceAddress* field set to 0xffffffffffffff, the *apsLinkKeyType* field set to

- 1342           0x01, the *LinkKey* field set to the distributed security global link key and  
1343           both the *OutgoingFrameCounter* and *IncomingFrameCounter* fields set to  
1344           0.
- 1345           18. The initiator SHALL then wait at least *bdbcTLMinStartupDelayTime*  
1346           seconds to allow the target to start the network.
- 1347           19. If the *logical type* field of the node descriptor for the initiator is not equal  
1348           to 0b010 (ZigBee end device) or a *network start request* inter-PAN  
1349           command frame was not sent, the initiator SHALL continue from step 26.
- 1350           20. The initiator SHALL perform a network rejoin request. To do this, the  
1351           initiator issues the *NLME-JOIN.request* primitive with the *ExtendedPANId*  
1352           parameter set to the extended PAN identifier of the selected network, the  
1353           *RejoinNetwork* parameter set to 0x02 (the node is joining the network  
1354           using the NWK rejoining procedure), the *ScanChannels* parameter set to  
1355           0x00000000, the *ScanDuration* parameter set to 0x00, the  
1356           *CapabilityInformation* set appropriately for the node and the  
1357           *SecurityEnable* parameter set to TRUE. On receipt of the *NLME-*  
1358           *JOIN.confirm* primitive from the NWK layer, the initiator is notified of the  
1359           status of the request for a network rejoin. The initiator SHALL then  
1360           continue from step 26.
- 1361           21. The initiator SHALL perform a network discovery to establish the network  
1362           parameters. To do this, the initiator issues the *NLME-NETWORK-*  
1363           *DISCOVERY.request* primitive to the NWK layer, with the *ScanChannels*  
1364           parameter set to *bdbcTLPrimaryChannelSet* and the *ScanDuration*  
1365           parameter set to *bdbScanDuration*. On receipt of the *NLME-NETWORK-*  
1366           *DISCOVERY.confirm* primitive from the NWK layer, the initiator is  
1367           notified of the results. Based on these results, the initiator SHALL select  
1368           suitable values for the logical channel, PAN identifier and extended PAN  
1369           identifier for the network.
- 1370           22. The initiator SHALL then copy the new network parameters to its network  
1371           information base and start operating on the new network. To do this, the  
1372           initiator issues the *NLME-START-ROUTER.request* primitive to the NWK  
1373           layer with the *BeaconOrder* parameter set to 0x0f, the *SuperframeOrder*  
1374           set to 0x00 and the *BatteryLifeExtension* parameter set to FALSE. On  
1375           receipt of the *NLME-START-ROUTER.confirm* primitive, the initiator is  
1376           notified of the status of the request to start.
- 1377           23. The initiator SHALL generate and unicast a *touchlink commissioning*  
1378           cluster *network join router request* or *network join end device* inter-PAN  
1379           command frame to the selected target, depending on whether the target is a  
1380           ZigBee router or a ZigBee end device, respectively, with the *extended PAN*  
1381           *identifier*, *network update identifier*, *logical channel* and *PAN identifier*  
1382           fields set to the corresponding network parameter values as used by the



1383 initiator. All other fields SHALL be specified according to sub-clause  
1384 8.7.1.

1385 24. The initiator SHALL then enable its receiver and wait for at most  
1386 *bdbcRxWindowDuration* seconds or until a corresponding response inter-  
1387 PAN command frame is received from the intended target with the same  
1388 *inter-PAN transaction identifier* field matching that used by the initiator in  
1389 its *scan request* command frame. The corresponding response to a  
1390 *network join router request* and a *network join end device request*  
1391 command frame is a *touchlink commissioning cluster network join router*  
1392 *response* and *network join end device response* command frame,  
1393 respectively. If a corresponding response inter-PAN command frame is  
1394 not received within *bdbcRxWindowDuration* seconds or if a corresponding  
1395 response inter-PAN command frame is received within  
1396 *bdbcRxWindowDuration* seconds but with a non-zero value in the *Status*  
1397 parameter, the initiator sets *bdbCommissioningStatus* to  
1398 TARGET\_FAILURE and it SHALL terminate the touchlink procedure for  
1399 an initiator.

1400 25. The initiator SHALL then wait at least *bdbcTLMinStartupDelayTime*  
1401 seconds to allow the target to start the network or to start operating on the  
1402 network correctly.

1403 26. If the initiator is a simple device, it SHALL establish binding links in the  
1404 binding table to the target. Conversely, if the initiator is not a simple  
1405 device, it MAY establish binding links in the binding table to the target. If  
1406 binding links are to be established, the initiator SHALL then, based on the  
1407 endpoint and device identifier information received in the *scan response*  
1408 and/or *device information response* inter-PAN command frames, establish  
1409 binding links in the binding table for matching client/server clusters on the  
1410 initiator and the corresponding server/client clusters on the target. The  
1411 initiator sets *bdbCommissioningStatus* to SUCCESS, sets  
1412 *bdbNodeIsOnANetwork* to TRUE and it SHALL terminate the touchlink  
1413 procedure for an initiator.

## 1414 8.7.1 General field settings for network start/join commands

### 1415 8.7.1.1 Inter-PAN transaction identifier field

1416 The *inter-PAN transaction identifier* field SHALL be set to the same value used in the  
1417 *scan request* command frame.

### 1418 8.7.1.2 Key index and encrypted network key fields

1419 The *key index* field SHALL be set to the touchlink key index (see [R2]) corresponding  
1420 to the key that was used to encrypt the ZigBee network key in the *encrypted network*  
1421 *key* field (i.e., the touchlink preconfigured link key). This value SHALL be set to  
1422 0x04 during certification testing or 0x0f at all other times.

1423 The *encrypted network key* field SHALL contain the encrypted ZigBee network key  
1424 that is to be used for securing the network. The ZigBee network key SHALL be  
1425 encrypted with the touchlink preconfigured link key.

#### 1426 **8.7.1.3 Network address field**

1427 The *network address* field SHALL be set to the network address with which the target  
1428 is to operate on the network.

1429 If the value of the *aplFreeNwkAddrRangeBegin* attribute (see [R2]) is equal to  
1430 0x0000 (initiator joined a network using MAC association), the address SHALL be  
1431 stochastically generated according to the classical ZigBee mechanism. If the value of  
1432 the *aplFreeNwkAddrRangeBegin* attribute is not equal to 0x0000, the address SHALL  
1433 be equal to *aplFreeNwkAddrRangeBegin* and then this value SHALL be incremented.

#### 1434 **8.7.1.4 Group identifiers begin/end fields**

1435 The *group identifiers begin* and *group identifiers end* fields SHALL be set to the  
1436 permissible range of group identifiers that are assigned to the target.

1437 If the target requested a set of group identifiers in its *scan response* command frame  
1438 and the value of the *aplFreeGroupIDAddrRangeBegin* attribute (see [R2]) is equal to  
1439 0x0000 (initiator joined a network using MAC association), the *group identifiers*  
1440 *begin* and *group identifiers end* fields SHALL be set to 0x0000. If the target  
1441 requested a set of group identifiers in its *scan response* command frame and the value  
1442 of the *aplFreeGroupIDAddrRangeBegin* attribute is not equal to 0x0000, a range of  
1443 group identifiers SHALL be allocated for the target and the *group identifiers begin*  
1444 and *group identifiers end* fields set accordingly.

#### 1445 **8.7.1.5 Free network/group address range begin/end fields**

1446 The *free network address range begin*, *free network address range end*, *free group*  
1447 *identifier range begin* and *free group identifier range end* fields SHALL be set to the  
1448 permissible range of network addresses and group identifiers that are assigned to the  
1449 target for future allocation to joining devices.

1450 If the target indicated that it was address assignment capable in its *scan response*  
1451 command frame and the value of the *aplFreeNwkAddrRangeBegin* attribute (see [R2])  
1452 is equal to 0x0000, the *free network address range begin*, *free network address range*  
1453 *end*, *free group identifier range begin* and *free group identifier range end* fields  
1454 SHALL be set to 0x0000. If the target indicated that it was address assignment  
1455 capable in its *scan response* command frame and the value of the  
1456 *aplFreeNwkAddrRangeBegin* attribute is not equal to 0x0000, a range of network  
1457 addresses and group identifiers SHALL be allocated for the target to use for its own  
1458 purposes and the *free network address range begin*, *free network address range end*,  
1459 *free group identifier range begin* and *free group identifier range end* fields set  
1460 accordingly.

### 1461 **8.8 Touchlink procedure for a target**

1462 This section defines the touchlink procedure for a target. In this procedure, the target  
1463 responds to touchlink requests from the initiator and either starts a new network or

1464 joins the network of the initiator. As this procedure is followed as a response to  
1465 touchlink requests from an initiator, it is not instigated via the top-level  
1466 commissioning procedure.

1467 The target SHALL NOT change its given network address unless it leaves the  
1468 network and joins another or if required to do so in order to resolve an address  
1469 conflict.

1470 If the target is a sleeping ZigBee end device it SHALL first need to be woken up by  
1471 some application means so that it can enable its receiver and respond to the scan from  
1472 the initiator.

1473 If the target receives an additional *touchlink commissioning cluster scan request*  
1474 command frame before the current transaction has completed, it MAY restart the  
1475 procedure again from the beginning or discard the frame.

1476 Note that simply accepting *touchlink commissioning cluster network start request* and  
1477 *network join router/end device request* command frames could lead to undesired  
1478 application behavior as the target leaves its current network and joins another network;  
1479 this is known in touchlink as *stealing*. For this reason, the procedure allows a target  
1480 to not accept these commands and indicate this by setting the *Status* field of the  
1481 corresponding *touchlink commissioning cluster network start response* or *network join*  
1482 *router/end device* command frame to indicate a failure.

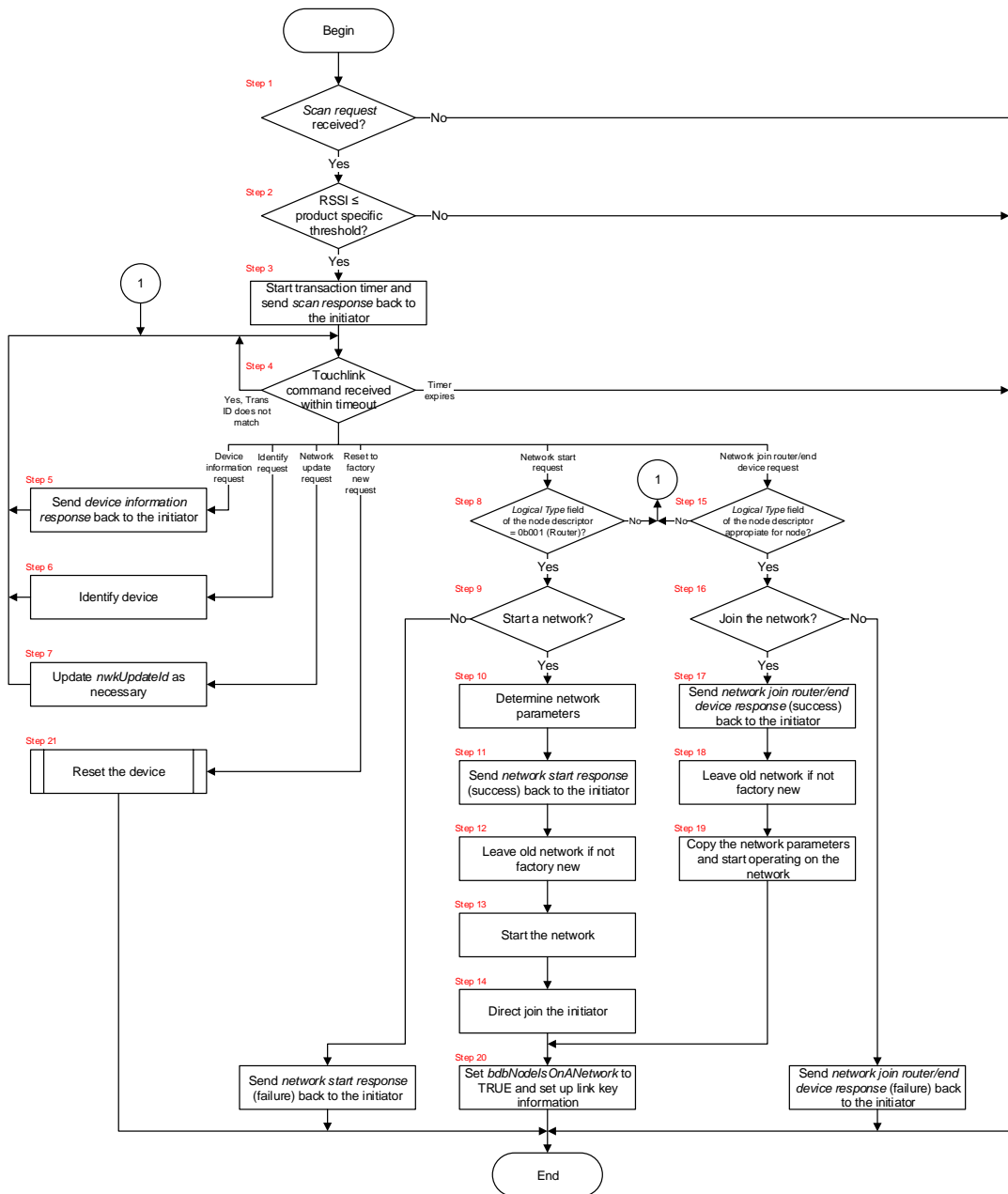
1483 The conditions under which the *network start request*, *network join router/end device*  
1484 *request* and also *network update request* command frames are or are not accepted is  
1485 (manufacturer) product specific. Here a balance can be made between security (e.g.,  
1486 not allowing the node to be stolen when part of a centralized security network) and  
1487 user friendliness (e.g., always allowing the node to be stolen) as different  
1488 requirements exist for both professional and consumer applications.

1489 A variable is defined for this procedure: a 32-bit unsigned integer value, *vIPTransID*,  
1490 which is used to store the *inter-PAN transaction identifier* field of the incoming  
1491 *touchlink commissioning cluster scan request* inter-PAN command frame.

1492 Figure 9 illustrates a simplified version of this procedure for quick reference.



1493



1494  
1495

1496  
1497

**Figure 9 – Touchlink procedure for a target**

1498  
1499  
1500  
1501  
1502  
1503  
1504

1. On receipt of a command other than the *touchlink commissioning cluster scan request* inter-PAN command frame, the target SHALL terminate the touchlink procedure for a target.
2. The target sets *vIPTransID* to the value of the *inter-PAN transaction identifier* field and it SHALL determine whether to respond. If the *scan request* command was received with an RSSI less than or equal to a certain product specific threshold or the *link initiator* sub-field of the *touchlink information*

- 1505 field is equal to 0, the target SHALL discard the frame and terminate the  
1506 touchlink procedure for a target.
- 1507 3. The target starts a timer for the current transaction to expire after  
1508 *bdbcTLInterPANTransIdLifetime* seconds. The target SHALL then generate  
1509 and unicast back to the initiator a *touchlink commissioning cluster scan*  
1510 *response* inter-PAN command frame as follows. The *inter-PAN transaction*  
1511 *identifier* field SHALL be set to *vIPTransID*. The *RSSI correction* field  
1512 SHALL be set to a product specific RSSI correction value in order to  
1513 compensate for RF signals losses between the radio and the outer side of a  
1514 product; the initiator can then use this value in combination with the RSSI  
1515 from each discovered target to select an appropriate target to continue with  
1516 touchlink commissioning. The *touchlink priority request* sub-field of the  
1517 *touchlink information* field SHALL be set to 1 if the target wishes to be  
1518 considered as a priority by the initiator during touchlinking (e.g. if the target is  
1519 power constrained and is responding to the scan following a button press from  
1520 the user). The *response identifier* field SHALL be set to a random (non-  
1521 sequential) value. If the *logical type* field of the node descriptor for the target  
1522 is equal to 0b001 (ZigBee router) and *bdbNodeIsOnANetwork* is equal to  
1523 TRUE, the *extended PAN identifier*, *network update identifier*, *logical*  
1524 *channel*, *PAN identifier* and *network address* fields SHALL be set to the  
1525 corresponding values of the network on which the target is currently operating.  
1526 If the *logical type* field of the node descriptor for the target is not equal to  
1527 0b001 (ZigBee router) or *bdbNodeIsOnANetwork* is equal to FALSE, the  
1528 *extended PAN identifier*, *network update identifier*, *logical channel*, *PAN*  
1529 *identifier* and *network address* fields SHALL be set to zero. All other fields  
1530 SHALL be set according to the specifics of the target.
- 1531 4. On receipt of a *touchlink commissioning cluster device information request*,  
1532 *identify request*, *network start request*, *network join router request*, *network*  
1533 *join end device request* or *reset to factory new request* inter-PAN command  
1534 frame with an *inter-PAN transaction identifier* field not equal to *vIPTransID*,  
1535 the target SHALL discard the frame and continue from step 4. If the  
1536 transaction timer expires, the target SHALL terminate the touchlink procedure  
1537 for a target.
- 1538 5. On receipt of a command other than the *device information request* inter-PAN  
1539 command frame, the target SHALL continue from step 6. The target SHALL  
1540 generate and unicast back to the initiator a *touchlink commissioning cluster*  
1541 *device information response* inter-PAN command frame as follows. The *inter-*  
1542 *PAN transaction identifier* field SHALL be set to *vIPTransID*. All other  
1543 fields SHALL be set according to the specifics of the target. The target  
1544 SHALL then continue from step 4.
- 1545 6. On receipt of a command other than the *identify request* inter-PAN command  
1546 frame, the target SHALL continue from step 8. The target SHALL identify

- 1547 itself in an application specific way (e.g., by flashing a lamp) according to the  
1548 value of the *identify time* field. No response SHALL be generated to an  
1549 *identify request* inter-PAN command frame. The identify operation SHALL  
1550 NOT block the target from receiving further commands. The target SHALL  
1551 then continue from step 4.
- 1552 7. On receipt of a command other than the *network update request* inter-PAN  
1553 command frame, the target SHALL continue from step 7. If the *extended PAN*  
1554 *identifier* and *PAN identifier* fields of the *network update request* inter-PAN  
1555 command frame are not identical to its stored values or the *network update*  
1556 *identifier* field is lower than or equal to *nwkUpdateId*, the target SHALL  
1557 discard the frame and continue from step 4. If the *extended PAN identifier* and  
1558 *PAN identifier* fields of the *network update request* inter-PAN command  
1559 frame are identical to its stored values and the *network update identifier* field  
1560 is higher than *nwkUpdateId*, the target SHALL update *nwkUpdateId* and its  
1561 current logical channel with the values of the *network update identifier* and  
1562 *logical channel* fields, respectively. The target SHALL then continue from  
1563 step 4.
- 1564 8. On receipt of a command other than the *network start request* inter-PAN  
1565 command frame, the target SHALL continue from step 15. If the *logical type*  
1566 field of the node descriptor is not equal to 0b001 (ZigBee router), the target  
1567 SHALL discard the frame and continue from step 4.
- 1568 9. The target SHALL decide by application specific means whether to allow  
1569 itself to start a new network. If the target decides not to start a new network, it  
1570 SHALL generate and unicast back to the initiator a *touchlink commissioning*  
1571 cluster *network start response* inter-PAN command frame with the *inter-PAN*  
1572 *transaction identifier* field set to *vIPTransID* and the *Status* field set to 0x01  
1573 (failure). The target SHALL then terminate the touchlink procedure for a  
1574 target.
- 1575 10. The target SHALL perform a network discovery to establish the network  
1576 parameters. To do this, the target issues the *NLME-NETWORK-*  
1577 *DISCOVERY.request* primitive to the NWK layer, with the *ScanChannels*  
1578 parameter set either to correspond to the single *logical channel* field of the  
1579 received *network start request* inter-PAN command frame if it is not equal to  
1580 zero or to *bdbcTLPrimaryChannelSet* if it is equal to zero and the  
1581 *ScanDuration* parameter set to *bdbScanDuration*. On receipt of the *NLME-*  
1582 *NETWORK-DISCOVERY.confirm* primitive from the NWK layer, the target is  
1583 notified of the results. Based on these results, the target SHALL select  
1584 suitable values for the logical channel, PAN identifier and extended PAN  
1585 identifier for the network.
- 1586 11. The target SHALL generate and unicast back to the initiator a *network start*  
1587 *response* inter-PAN command frame as follows. The *inter-PAN transaction*  
1588 *identifier* field SHALL be set to *vIPTransID*. The *Status* field SHALL be set

- 1589 to 0x00 (success). All other fields SHALL be set as appropriate to the verified  
1590 network parameters.
- 1591 12. If *bdbNodeIsOnANetwork* is equal to TRUE, the target SHALL perform a  
1592 leave request on its old network. To do this, the target issues the *NLME-*  
1593 *LEAVE.request* primitive to the NWK layer with the *DeviceAddress* parameter  
1594 set to NULL, the *RemoveChildren* parameter set to FALSE and the *Rejoin*  
1595 parameter set to FALSE. On receipt of the *NLME-LEAVE.confirm* primitive,  
1596 the target is notified of the status of the request to leave the network. The  
1597 target SHALL then clear all ZigBee persistent data (see sub-clause 6.9) except  
1598 the outgoing NWK frame counter.
- 1599 13. The target SHALL then copy the new network parameters to its network  
1600 information base and start operating on the new network. To do this, the  
1601 target issues the *NLME-START-ROUTER.request* primitive to the NWK layer  
1602 with the *BeaconOrder* parameter set to 0x0f, the *SuperframeOrder* set to 0x00  
1603 and the *BatteryLifeExtension* parameter set to FALSE. On receipt of the  
1604 *NLME-START-ROUTER.confirm* primitive, the target is notified of the status  
1605 of the request to start.
- 1606 14. The target SHALL perform a direct join on behalf of the initiator. To do this,  
1607 the target issues the *NLME-DIRECT-JOIN.request* primitive to the NWK layer  
1608 with the *DeviceAddress* parameter set to the IEEE address of the initiator. On  
1609 receipt of the *NLME-DIRECT-JOIN.confirm* primitive, the target is notified of  
1610 the status of the direct join request. The target SHALL then continue from  
1611 step 20.
- 1612 15. On receipt of a command other than the *network join router request* or a  
1613 *network join end device* inter-PAN command frame, the target SHALL  
1614 continue from step 21. If a *network join router request* inter-PAN command  
1615 frame was received and the *logical type* field of the node descriptor is not  
1616 equal to 0b001 (ZigBee router) or a *network join end device* inter-PAN  
1617 command frame was received and the *logical type* field of the node descriptor  
1618 is not equal to 0b010 (ZigBee end device), the target SHALL discard the  
1619 frame and continue from step 4.
- 1620 16. The target SHALL decide by application specific means whether to allow  
1621 itself to be joined to another network. If the target decides not to be joined to  
1622 another network, it SHALL generate and unicast back to the initiator a  
1623 corresponding *touchlink commissioning cluster network join router response*  
1624 or *network join end device response* inter-PAN command frame, depending on  
1625 whether a *network join router request* or *network join end device request* inter-  
1626 PAN command frame, respectively, was received with the *inter-PAN*  
1627 *transaction identifier* field set to *vIPTransID* and the *Status* field set to 0x01  
1628 (failure). The target SHALL then terminate the touchlink procedure for a  
1629 target.

- 1630 17. The target SHALL generate and unicast back to the initiator a *touchlink*  
1631 *commissioning cluster network join router response* or *network join end device*  
1632 *response* inter-PAN command frame, depending on whether a *network join*  
1633 *router request* or *network join end device request* inter-PAN command frame,  
1634 respectively, was received with the *inter-PAN transaction identifier* field set to  
1635 *vIPTransID* and the *Status* field set to 0x00 (success). The target sets  
1636 *bdbNodeJoinLinkKeyType* to 0x03 (touchlink preconfigured link key).
- 1637 18. If *bdbNodeIsOnANetwork* is equal to TRUE, the target SHALL perform a  
1638 leave request on its old network. To do this, the target issues the *NLME-*  
1639 *LEAVE.request* primitive to the NWK layer with the *DeviceAddress* parameter  
1640 set to NULL, the *RemoveChildren* parameter set to FALSE and the *Rejoin*  
1641 parameter set to FALSE. On receipt of the *NLME-LEAVE.confirm* primitive,  
1642 the target is notified of the status of the request to leave the network. The  
1643 target SHALL then clear all ZigBee persistent data (see sub-clause 6.9) except  
1644 the outgoing NWK frame counter.
- 1645 19. The target SHALL then copy the new network parameters to its network  
1646 information base. If the *logical type* field of the node descriptor is equal to  
1647 0b010 (ZigBee end device), the target SHALL continue from step 20. The  
1648 target issues the *NLME-START-ROUTER.request* primitive to the NWK layer  
1649 with the *BeaconOrder* parameter set to 0x0f, the *SuperframeOrder* set to 0x00  
1650 and the *BatteryLifeExtension* parameter set to FALSE. On receipt of the  
1651 *NLME-START-ROUTER.confirm* primitive, the target is notified of the status  
1652 of the request to start.
- 1653 20. The target sets *bdbNodeIsOnANetwork* to TRUE, sets *apsTrustCenterAddress*  
1654 to 0xffffffffffffff and it SHALL determine whether an entry exists in  
1655 *apsDeviceKeyPairSet* with a *DeviceAddress* field which corresponds to  
1656 0xffffffffffffff. If such an entry does not exist, the target SHALL create a  
1657 new entry with the *DeviceAddress* field set to 0xffffffffffffff, the  
1658 *apsLinkKeyType* field set to 0x01, the *LinkKey* field set to the distributed  
1659 security global link key and both the *OutgoingFrameCounter* and  
1660 *IncomingFrameCounter* fields set to 0. The target SHALL then terminate the  
1661 touchlink procedure for a target.
- 1662 21. On receipt of a command other than the *reset to factory new request* inter-  
1663 PAN command frame, the target SHALL discard the command and continue  
1664 from step 4. The target SHALL follow the touchlink reset procedure (see sub-  
1665 clause 9.2) and then terminate the touchlink procedure for a target.



## 1666 **9 Reset**

1667 A node implementation SHALL provide an interactive mechanism to reset itself to its  
1668 factory settings. This mechanism SHALL be accessible to the installer of the product.

1669 ZigBee-PRO provides several mechanisms for reset with various levels of impact  
1670 from just resetting the application cluster attributes to clearing ZigBee persistent data  
1671 (such as network settings, groups and bindings) and leaving the network. All reset  
1672 mechanisms SHALL preserve the single outgoing NWK frame counter, maintained  
1673 by all devices.<sup>3</sup>

### 1674 **9.1 Reset via the basic cluster**

1675 The *basic* cluster provides a *reset to factory defaults* command which is designed to  
1676 only reset the attributes of all clusters supported on a target device to their default  
1677 settings, i.e., network settings, groups and bindings are not affected by this command.

1678 To reset all attributes on a target device to their default values using the *basic* cluster,  
1679 an initiator device SHALL generate and transmit to the intended target device a *basic*  
1680 cluster, *reset to factory defaults* command.

1681 On receipt of the *basic* cluster, *reset to factory defaults* command, the target device  
1682 SHALL reset the attributes of all clusters supported on the target device to their  
1683 default values. All other values such as network settings, frame counters, groups and  
1684 bindings SHALL be preserved.

### 1685 **9.2 Reset via the touchlink commissioning cluster**

1686 The *touchlink commissioning* cluster provides a *reset to factory new request*  
1687 command which is designed to clear all ZigBee persistent data (see sub-clause 6.9),  
1688 except the outgoing NWK frame counter, and perform a reset such that the target is in  
1689 much the same state as it was when it left the factory. This command SHALL be  
1690 transmitted via inter-PAN communication. Note that as this command is transmitted  
1691 using inter-PAN communication, security is not used.

1692 To reset a target to its factory new state using the *touchlink commissioning* cluster, an  
1693 initiator SHALL first follow the first 12 steps of the touchlink procedure for an  
1694 initiator (see sub-clause 8.7) with an extended channel scan. The initiator SHALL  
1695 then generate and transmit to the intended target a *touchlink commissioning* cluster,  
1696 *reset to factory new request* inter-PAN command frame.

1697 On receipt of the *touchlink commissioning* cluster, *reset to factory new request* inter-  
1698 PAN command frame and if the target is on a centralized security network (i.e.,  
1699 *apsTrustCenterAddress* is not equal to 0xffffffffffff), the target MAY, under  
1700 product specific conditions, discard the frame and perform no further processing.

1701 On receipt of the *touchlink commissioning* cluster, *reset to factory new request* inter-  
1702 PAN command frame with an invalid transaction identifier (i.e., the frame was not

---

<sup>3</sup> The single frame counter SHALL only be reset in the cases specified in ZigBee-PRO, revision 21 or higher (see [R1]).

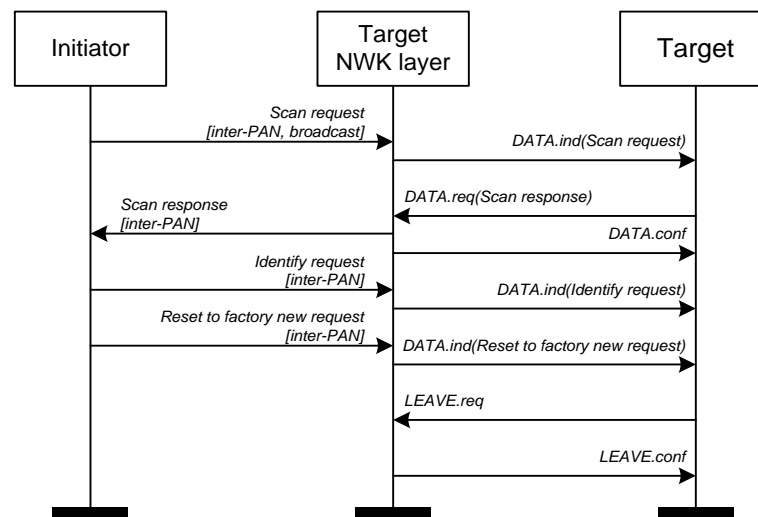
1703 received within the current active transaction), the target SHALL discard the frame  
 1704 and perform no further processing.

1705 On receipt of the *touchlink commissioning* cluster, *reset to factory new request* inter-  
 1706 PAN command frame with a valid transaction identifier, i.e., immediately following a  
 1707 touchlink device discovery, the target SHALL perform a leave request on the  
 1708 network. To do this, the target issues the *NLME-LEAVE.request* primitive to the  
 1709 NWK layer with the *DeviceAddress* parameter set to NULL, the *RemoveChildren*  
 1710 parameter set to FALSE and the *Rejoin* parameter set to FALSE. On receipt of the  
 1711 *NLME-LEAVE.confirm* primitive, the target is notified of the status of the request to  
 1712 leave the network.

1713 The target SHALL then clear all ZigBee persistent data (see sub-clause 6.9) except  
 1714 the outgoing NWK frame counter.

1715 The sequence of events for resetting a target to factory new via the *touchlink*  
 1716 *commissioning* cluster is illustrated in Figure 10.

1717



1718

1719 **Figure 10 – Resetting a target to factory new via the *touchlink***  
 1720 ***commissioning* cluster**

1721

### 1722 9.3 Reset via the network leave command

1723 ZigBee-PRO provides a network *leave* command which is designed to request that a  
 1724 remote node leaves the network by clearing all ZigBee persistent data (see sub-clause  
 1725 6.9), except the outgoing NWK frame counter, and perform a reset such that the node  
 1726 is in much the same state as it was when it left the factory.

1727 The network *leave* command is specified in sub-clause 3.4.4 of [R1] and its use is  
 1728 specified in sub-clause 3.6.1.10 of [R1].

### 1729 9.4 Reset via *Mgmt\_Leave\_req* ZDO command

1730 ZigBee-PRO provides an *Mgmt\_Leave\_req* ZDO command which is designed to  
 1731 request that a remote node leaves the network by clearing all ZigBee persistent data

1732 (see sub-clause 6.9), except the outgoing NWK frame counter, and perform a reset  
1733 such that the node is in much the same state as it was when it left the factory.

1734 The *Mgmt\_Leave\_req* ZDO command is specified in sub-clause 2.4.3.3.5 of [R1].

## 1735 **9.5 Reset via a local action**

1736 It is RECOMMENDED that a local action be provided to allow a node to be reset  
1737 such that all ZigBee persistent data (see sub-clause 6.9), except the outgoing NWK  
1738 frame counter, is cleared and a reset is performed such that the node is in much the  
1739 same state as it was when it left the factory.

1740 This local action SHOULD be invoked via some user accessible implementation  
1741 specific application stimulus, such as an external button press on the node or through  
1742 some software activation. It is RECOMMENDED to only allow this procedure to be  
1743 activated if the user is physically present at the node.

1744 If a node receives some stimulus from the application to reset and leave its current  
1745 network, it SHALL perform a leave request on the network. To do this, the node  
1746 issues the *NLME-LEAVE.request* primitive to the NWK layer with the *DeviceAddress*  
1747 parameter set to NULL, the *RemoveChildren* parameter set to FALSE and the *Rejoin*  
1748 parameter set to FALSE. On receipt of the *NLME-LEAVE.confirm* primitive, the  
1749 node is notified of the status of the request to leave the network.

1750 The node SHALL then clear all ZigBee persistent data (see sub-clause 6.9) except the  
1751 outgoing NWK frame counter.



## 1752 10 Security

### 1753 10.1 Install codes

1754 This section describes the out of band process for establishing pre-configured Trust  
 1755 Center link keys, the format of the Install Code required, and the hashing function  
 1756 used to derive the pre-configured link key from the Install Code. Note that Install  
 1757 Codes SHALL be random but MAY NOT be unique.

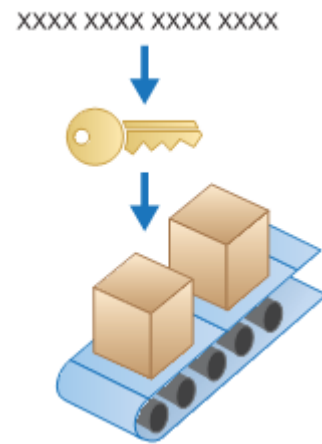
1758 As portrayed in Figure 11, during the manufacturing process a random Install Code is  
 1759 created for each of the nodes. This Install Code is provided for the node in a  
 1760 manufacturer-specific way (labeling, etc.) and referred to during installation. The  
 1761 space of Install Codes SHOULD possess the same randomness properties as a key  
 1762 space. Knowing a set of Install Codes SHOULD NOT yield any knowledge of another  
 1763 Install Code and each Install Code SHOULD be equally probable.

1764

Step 1: An Install Code is created and made available.

Step 2: The pre-configured link key is derived from the Install Code using the Matyas-Meyer-Oseas hash function.

Step 3: The pre-configured link key is configured in the node.



1765 **Figure 11 – Node Install Code process**

1766

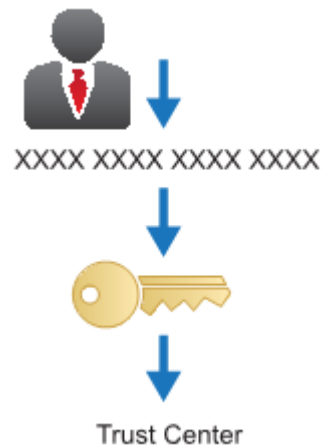
1767 As portrayed in Figure 12, during the installation process the initial Trust Center link  
 1768 key is derived from the Install Code and sent via an out of band communication  
 1769 channel to the Trust Center. The Trust Center uses this key as the Trust Center link  
 1770 key which is subsequently used to configure the network key of the associating node.

1771

Step 1: The Install Code is sent out of band.

Step 2: The pre-configured link key is derived from the Install Code using the Matyas-Meyer-Oseas hash function.

Step 3: The pre-configured link key is installed in the Trust Center.



1772 **Figure 12 – Install code use with the Trust Center**

1773

### 1774 10.1.1 Install code format

1775 The Install Code consists of a 128 bit number and a 16 bit CRC (using CCITT CRC  
1776 standard polynomial:  $x^{16} + x^{12} + x^5 + 1$ ). When printed or displayed, Install Codes  
1777 are represented as multiple groups of 4 hexadecimal digits.

1778 Example:

1779 Install code of “83FE D340 7A93 9723 A5C6 39B2 6916 D505 C3B5”

1780 Where values 0x83, 0xFE, 0xD3, 0x40, 0x7A, 0x93, 0x97, 0x23, 0xA5, 0xC6, 0x39,  
1781 0xB2, 0x69, 0x16, 0xD5, and 0x05 are used to calculate the CRC16 with the result  
1782 returning 0xB5C3. (Note that the CRC16 and the install code itself are represented in  
1783 little endian byte order in the above example.)

#### 1784 10.1.1.1 CRC algorithm information

1785 As stated earlier, the Install Code CRC calculation is based upon the CRC 16-CCITT  
1786 algorithm and uses the following parameters:

1787 Length: 16

1788 Polynomial:  $x^{16} + x^{12} + x^5 + 1$  (0x1021)

1789 Initialization method: Direct

1790 Initialization value: 0xFFFF

1791 Final XOR value: 0xFFFF

1792 Reflected In: True

1793 Reflected Out: True

1794 Open source implementations of the CRC 16-CCITT algorithm are available on the  
1795 internet at sites like SourceForge and others. The source code is also available in [R5].

## 1796 **10.1.2 Hashing Function**

1797 An AES-128 key is derived from the Install Code using the Matyas-Meyer-Oseas  
1798 (MMO) hash function (See [R1], Annex B.6 with a digest size (hashlen) equal to 128  
1799 bits).

1800 Install code example:

1801 MMO hash applied to the Install Code “83FE D340 7A93 9723 A5C6 39B2 6916  
1802 D505” produces the key “66B6900981E1EE3CA4206B6B861C02BB”.

1803 Note: Least significant byte is 0x83 and most significant byte is 0x05.

### 1804 **10.1.2.1 MMO hash code example**

1805 Open source implementations of the MMO Hash based on the Rijndael  
1806 implementation are available on the internet at sites like SourceForge and others. The  
1807 source code is also available in [R5].

## 1808 **10.2 Node operations**

1809 Nodes joining the network SHALL also have policies that dictate what security they  
1810 expect from the network. The following are the settings that MAY be used to adjust  
1811 their security policy.

### 1812 **10.2.1 Joining node policy values**

1813 A joining node MAY have a set of policy values, for example if it is to be  
1814 commissioned into a network. However, it normally sets these policy values based on  
1815 whether it joins a centralized security network or a distributed security network. All  
1816 nodes except those designated as a ZigBee coordinator SHALL support joining  
1817 networks using either security model.

#### 1818 **10.2.1.1 *acceptNewUnsolicitedTrustCenterLinkKey* policy**

1819 This boolean indicates whether the node will accept a new, unsolicited APS transport  
1820 key message containing a Trust Center link key.

1821 Note this value is ignored in a distributed security network.

#### 1822 **10.2.1.2 *acceptNewUnsolicitedApplicationLinkKey* policy**

1823 This boolean indicates whether the node will accept a new unsolicited application link  
1824 key sent to it by the Trust Center or another device.

1825 This value MAY be used in distributed security networks if the device requires use of  
1826 APS encryption with a partner node.

### 1827 **10.2.2 Trust Center address**

1828 A node MAY know the address of the Trust Center prior to joining; this is dependent  
1829 upon the commissioning procedure for the node. If the Trust Center address is known  
1830 prior to the node joining the network then the commissioning procedure SHALL set  
1831 *apsTrustCenterAddress* to the value of the IEEE address of the Trust Center in the  
1832 network it will join.

1833 In most cases the network that the node will be joining is not known ahead of time.  
1834 Therefore it is RECOMMENDED that the commissioning process for a node not  
1835 preprogram the Trust Center address. In this case, the *apsTrustCenterAddress*  
1836 SHALL initially be set to 0xffffffffffffff. Once the node joins the network and  
1837 receives and decrypts the APS command transport key command containing the  
1838 network key, it SHALL set *apsTrustCenterAddress* to the value of the source address  
1839 in the command.

1840 If *bdbNodeIsOnANetwork* is equal to TRUE and *apsTrustCenterAddress* is equal to  
1841 0xffffffffffffff, the device has joined a distributed security network and the node  
1842 settings SHOULD be adjusted accordingly. Conversely, if *apsTrustCenterAddress* is  
1843 not equal to 0xffffffffffffff, the node has joined a centralized security network.

1844 For all subsequently received Trust Center or security related APS command frames  
1845 where a source address field is present, if *apsTrustCenterAddress* is not equal to  
1846 0xffffffffffffff then the node SHALL compare the value of *apsTrustCenterAddress*  
1847 with the source address value of the APS command. If the values do not match the  
1848 frame SHALL be dropped and no further processing SHALL take place.

### 1849 **10.2.3 Trust Center Link Keys**

1850 All nodes SHALL have an updated Trust Center link key once they are joined to a  
1851 centralized security network. This allows the use of secure communication for  
1852 notifications of joining events and for distributing network keys to devices that missed  
1853 key updates. Nodes SHALL use a preconfigured key to join the network and then  
1854 request an updated link key once joining is complete. Once the node has obtained an  
1855 updated trust-center link key it SHALL ignore any APS commands from the Trust  
1856 Center that are not encrypted with that key.

### 1857 **10.2.4 Requesting a Link Key**

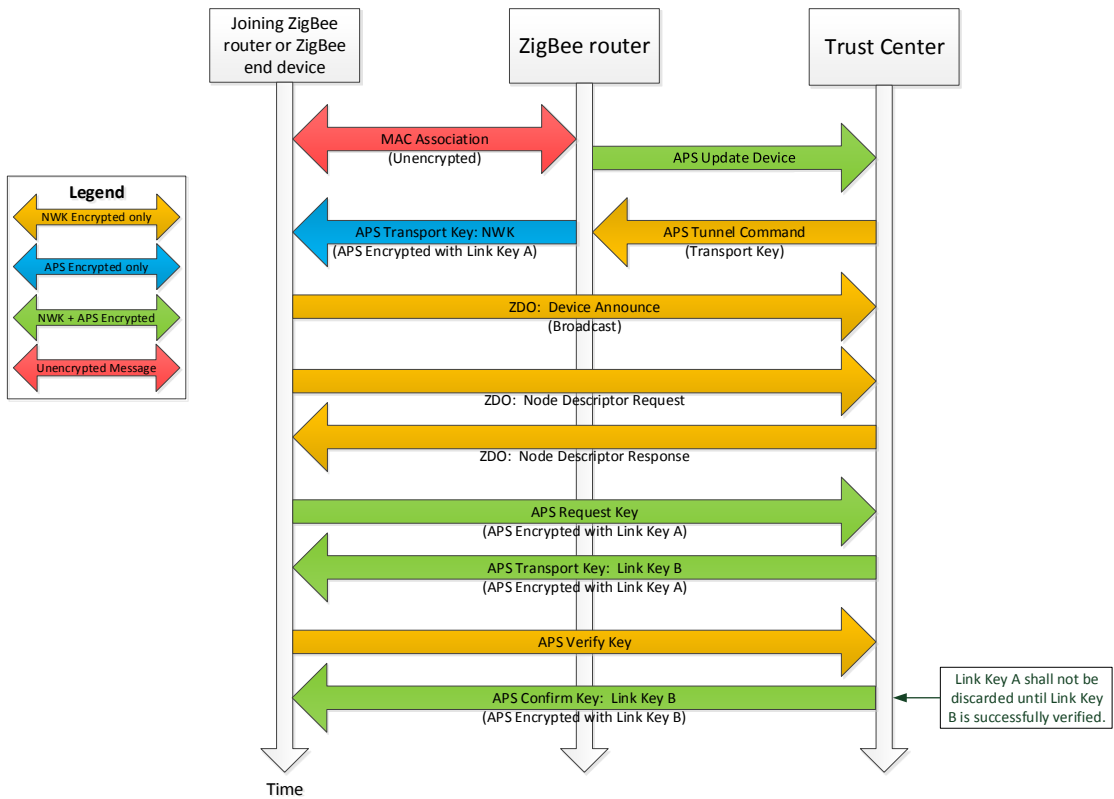
1858 If *bdbTCLinkKeyExchangeMethod* is equal to 0x00, the node SHALL exchange its  
1859 initial link key with one generated by the Trust Center as part of its initial joining  
1860 operations in a centralized security network.

1861 If *bdbTCLinkKeyExchangeMethod* is not equal to 0x00, the node SHALL follow the  
1862 appropriate procedure specified by this attribute. However, if the procedure fails, the  
1863 node SHALL fall back to the above link key exchange method 0x00. If this method is  
1864 successful, the node MAY treat the key as unauthorized for the purposes of allowing  
1865 access to restricted clusters.

### 1866 **10.2.5 Trust Center link key exchange procedure**

1867 This section defines the procedure to retrieve a new Trust Center link key for a node.  
1868 A sequence chart for this procedure showing the messages exchanged and the  
1869 corresponding keys used to encrypt the messages is illustrated in Figure 13.

1870



1871

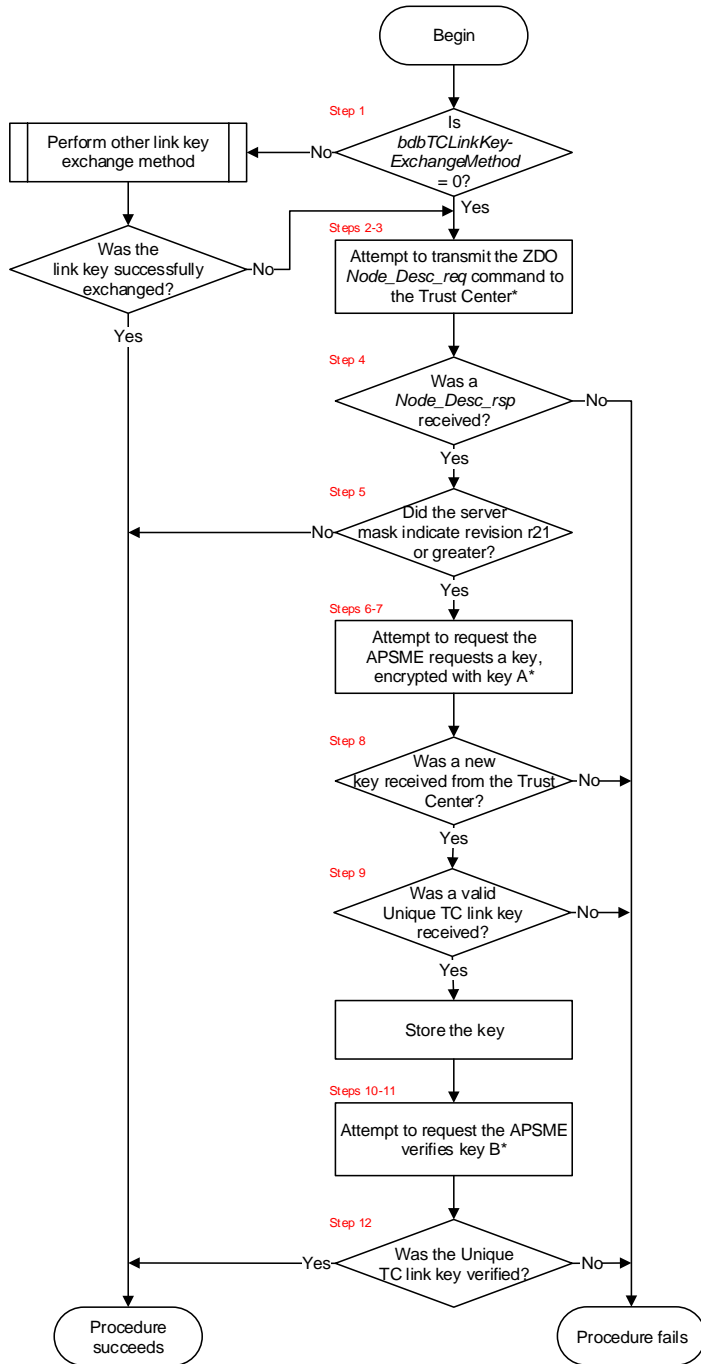
1872 **Figure 13 – Trust Center link key exchange procedure sequence chart**

1873

1874 Figure 14 illustrates a simplified version of this procedure for quick reference.

1875

1876



\* The node SHALL make an attempt and wait for up to *bdbTCLinkKeyExchangeTimeout* for a response. If no response is received before this timeout expires, the node shall repeat the attempt such that at most *bdbTCLinkKeyExchangeAttempts* are made in total. If no response is received after *bdbTCLinkKeyExchangeAttempts* are made, the attempt is considered to have failed.

1877

1878

**Figure 14 – Trust Center link key exchange procedure**

1879

1880

1881

1882

1883

1884

1885

1. The joining node SHALL examine its *bdbTCLinkKeyExchangeMethod*. If the *bdbTCLinkKeyExchangeMethod* is set to 0, then it SHALL continue from step 2. If the *bdbTCLinkKeyExchangeMethod* is set to another value, it SHALL execute the appropriate steps as defined by that mechanism. If the mechanism is successful, the node SHALL terminate the Trust Center link key exchange procedure with a success status.

- 1886 2. The joining node sets *bdbTCLinkKeyExchangeAttempts* to 0.
- 1887 3. The joining node SHALL send a *ZDO Node\_Desc\_req* command to the Trust
- 1888 Center. It then starts a timer of *bdbcTCLinkKeyExchangeTimeout* seconds and
- 1889 increments *bdbTCLinkKeyExchangeAttempts* by 1.
- 1890 4. If a *ZDO Node\_Desc\_rsp* command is not received before the timer expires,
- 1891 the joining node SHALL determine whether to retry the attempt as follows:
- 1892 a. If *bdbTCLinkKeyExchangeAttempts* is less than *bdbTCLinkKey-*
- 1893 *ExchangeAttemptsMax*, the joining node SHALL continue from step 3.
- 1894 b. If *bdbTCLinkKeyExchangeAttempts* is equal to *bdbTCLinkKey-*
- 1895 *ExchangeAttemptsMax* the joining node SHALL terminate the Trust
- 1896 Center link key exchange procedure with a failure status.
- 1897 5. If the *server\_mask* field of the receiver *node* descriptor indicates a stack
- 1898 revision of r20 or earlier, the joining node SHALL terminate the Trust Center
- 1899 link key exchange procedure with a success status.
- 1900 6. The joining node sets *bdbTCLinkKeyExchangeAttempts* to 0.
- 1901 7. The joining node SHALL request a new link key from the Trust Center. To do
- 1902 this, the joining node issues an *APSME-REQUEST-KEY.request* primitive
- 1903 encrypted with its initial Trust Center link key (key A). It then starts a timer
- 1904 of *bdbcTCLinkKeyExchangeTimeout* seconds and increment
- 1905 *bdbTCLinkKeyExchangeAttempts* by 1.
- 1906 8. If the joining node does not receive an *APSME-TRANSPORT-KEY.indication*
- 1907 primitive before the timer expires, the joining node SHALL determine
- 1908 whether to retry the attempt as follows:
- 1909 a. If *bdbTCLinkKeyExchangeAttempts* is less than *bdbTCLinkKey-*
- 1910 *ExchangeAttemptsMax*, the joining node SHALL continue from step 7.
- 1911 b. If *bdbTCLinkKeyExchangeAttempts* is equal to *bdbTCLinkKey-*
- 1912 *ExchangeAttemptsMax*, the joining node SHALL terminate the Trust
- 1913 Center link key exchange procedure with a failure status.
- 1914 9. The joining node SHALL find the entry in the *apsDeviceKeyPairSet* with a
- 1915 *DeviceAddress* that corresponds to the *apsTrustCenterAddress*. If the
- 1916 *KeyType* parameter of the received *APSME-TRANSPORT-KEY.indication*
- 1917 primitive is not equal to 0x04 (Unique Trust Center Link Key) or the link key
- 1918 contained in the primitive is identical to the *LinkKey* value of the
- 1919 *apsDeviceKeyPairSet* entry, the joining node SHALL terminate the Trust
- 1920 Center link key exchange procedure with a failure status. Otherwise, the
- 1921 joining node SHALL replace the *LinkKey* value with the key contained in the
- 1922 primitive (link key B), it MAY then set *OutgoingFrameCounter* to 0 and it
- 1923 SHALL set the *IncomingFrameCounter* to 0 for the *apsDeviceKeyPairSet*
- 1924 entry.
- 1925 10. The joining node sets *bdbTCLinkKeyExchangeAttempts* to 0.
- 1926 11. The joining node SHALL verify the new link key with the Trust Center. To
- 1927 do this, the joining node issues an *APSME-VERIFY-KEY.request* primitive to



- 1928           verify the new key (link key B). It then starts a timer of *bdbTCLink-*  
 1929           *KeyExchangeTimeout* seconds and increment *bdbTCLinkKeyExchange-*  
 1930           *Attempts* by 1.
- 1931       12. If the joining node does not receive an *APSME-CONFIRM-KEY.indication*  
 1932       primitive before the timer expires, the joining node SHALL determine  
 1933       whether to retry the attempt as follows:
- 1934           a. If *bdbTCLinkKeyExchangeAttempts* is less than *bdbTCLinkKey-*  
 1935           *ExchangeAttemptsMax*, the joining node SHALL continue from step  
 1936           11.
- 1937           b. If *bdbTCLinkKeyExchangeAttempts* is equal to *bdbTCLinkKey-*  
 1938           *ExchangeAttemptsMax*, the joining node SHALL terminate the Trust  
 1939           Center link key exchange procedure with a failure status.
- 1940       13. The joining node SHALL terminate the Trust Center link key exchange  
 1941       procedure with a success status.

1942       Note that the joining node SHALL consider Link key A to be valid until Link key B is  
 1943       successfully verified with the Trust Center with a successfully decrypted response.

#### 1944   **10.2.6 Receiving new Link Keys**

1945       It is possible the security policy of a node MAY restrict application link keys sent to it  
 1946       by the Trust Center. This could be because the node wishes to control which other  
 1947       nodes it shares link keys with, or because it uses some other mechanism to establish  
 1948       application link keys.

1949       There are instances where higher level application policies determine what data is  
 1950       shared with application link keys, for example, networks where updated Trust Center  
 1951       link keys are established through the Certificate Based Key Exchange protocol.

1952       If the node receives a transport key command containing a Trust Center link key, but  
 1953       it has not sent a request for one and *acceptNewUnsolicitedTrustCenterLinkKey* is set  
 1954       to FALSE, it SHALL ignore the message. If the node receives a transport key  
 1955       command containing an application link key, but it has not sent a request for one, and  
 1956       *acceptNewUnsolicitedApplicationLinkKey* is set to FALSE, it SHALL ignore the  
 1957       message.

### 1958   **10.3 Trust Center behavior**

#### 1959   **10.3.1 Adding the install code**

- 1960       1. Via some manufacturer specific means, the Trust Center SHALL decide  
 1961       whether to allow the node to join (see sub-clause 4.7.3 of [R1])
- 1962           a. If the node is *not* allowed to join, no further action is taken.
- 1963       2. The Trust Center then SHALL decide whether that joining node SHALL use  
 1964       the default link key or an installation code link key, as specified by  
 1965       *bdbJoinUsesInstallCodeKey*.
- 1966           a. If the Trust Center requires the use of installation code link keys then it  
 1967       SHALL add an entry into its AIB *apsDeviceKeyPairSet* with the

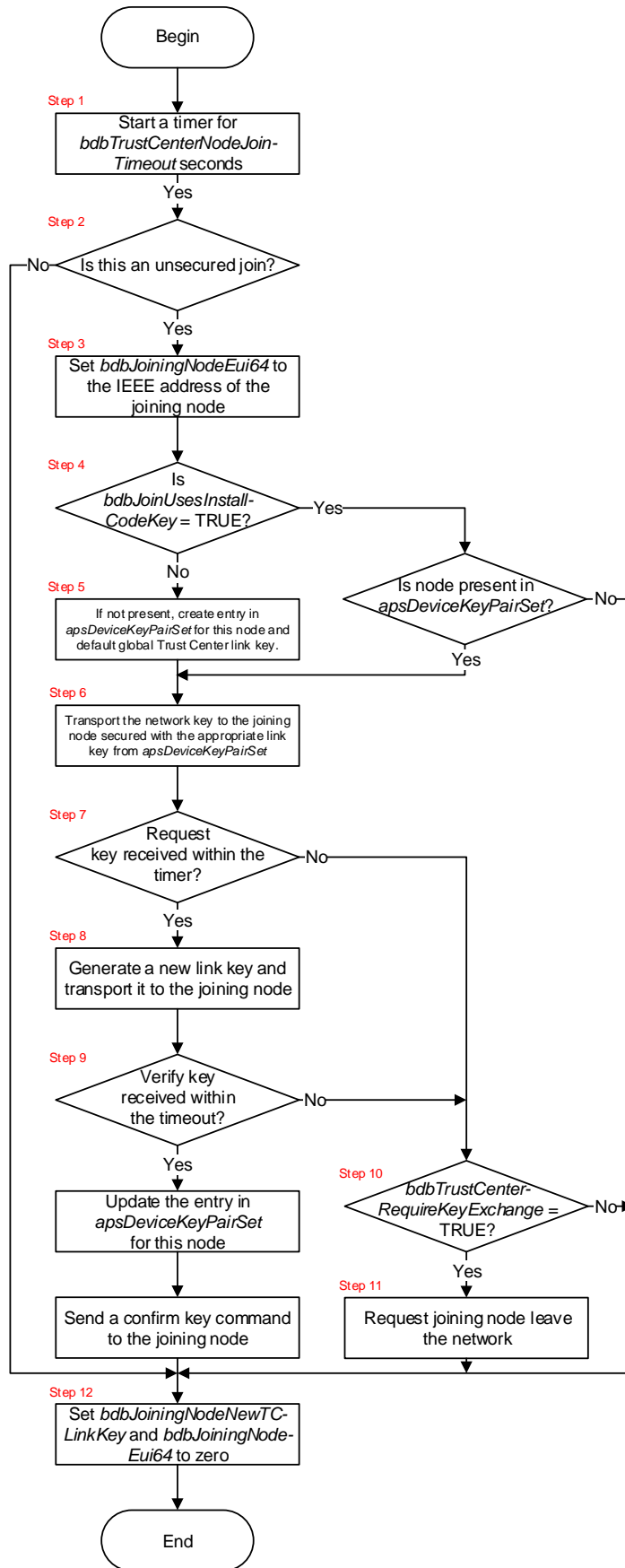


- 1968                    *DeviceAddress* set to the EUI64 of the joining node and the *LinkKey*  
1969                    value equal to the installation code link key.
- 1970                    i. The *apsLinkKeyType* of that entry SHALL be set to 0x00  
1971                    (Unique). See Table 4.39 in [R1].
  - 1972                    b. If the Trust Center does not require use of installation code link key  
1973                    then it shall create a corresponding entry in its AIB  
1974                    *apsDeviceKeyPairSet* when the node joins the network.

### 1975    **10.3.2 Adding a new node into the network**

1976    When the Trust Center is accepting a new node for joining it MAY choose whether  
1977    that node SHALL use the default Trust Center link key or an installation code key to  
1978    encrypt the network key. It MAY also choose to allow a mix of devices in the  
1979    network. This is per the policies of the Trust Center. This procedure describes how  
1980    the Trust Center will handle a node joining where the value of  
1981    *bdbTCLinkKeyExchangeMethod* is equal to 0x00 (APS Request Key establishment  
1982    method). Other values of *bdbTCLinkKeyExchangeMethod* are not yet supported.

1983    Figure 15 illustrates a simplified version of this procedure for quick reference.  
1984



1985

1986

Figure 15 – Trust Center link key exchange procedure

1987

1988

1989

1990

1991

1992

1993

1994

1995

1996

1997

1998

1999

2000

2001

2002

2003

2004

2005

2006

2007

2008

2009

2010

2011

2012

2013

2014

2015

2016

2017

2018

2019

2020

2021

2022

2023

2024

2025

2026

2027

2028

1. Upon receipt of an *APSME-UPDATE-DEVICE.indication* primitive from the APSME, the Trust Center SHALL start a timer for *bdbTrustCenterNodeJoinTimeout* seconds.
2. The Trust Center SHALL determine if the *Status* parameter is equal to 0x01 (Unsecured join).
  - a. If this is not true, the Trust Center SHALL continue from step 12.
3. The Trust Center SHALL set *bdbJoiningNodeEui64* to the *DeviceAddress* parameter in the *APSME-UPDATE-DEVICE.indication* primitive.
4. If *bdbJoinUsesInstallCodeKey* is equal to TRUE and *bdbJoiningNodeEui64* does not correspond to an entry in *apsDeviceKeyPairSet*, the Trust Center SHALL continue from step 12.
5. If *bdbJoinUsesInstallCodeKey* is equal to FALSE and *bdbJoiningNodeEui64* does not correspond to an entry in *apsDeviceKeyPairSet*, the Trust Center SHALL add an entry into its AIB *apsDeviceKeyPairSet* with the *DeviceAddress* parameter set to *bdbJoiningNodeEui64* and the *LinkKey* value set to the default global Trust Center link key (“ZigBeeAlliance09”).
  - a. The *apsLinkKeyType* of that entry SHALL be set to 0x01 (Global). See Table 4.39 in [R1].
6. The Trust Center SHALL transport the network key to the joining node by issuing the *APSME-TRANSPORT-KEY.request* primitive to the APSME encrypted with the *LinkKey* value of the *apsDeviceKeyPairSet* entry corresponding to the joining node.
7. If, within the timeout initiated in step 1, an *APSME-REQUEST-KEY.indication* primitive with an IEEE address equal to *bdbJoiningNodeEui64* is not received from the APSME, the Trust Center SHALL continue from step 10.
8. The Trust Center SHALL generate a link key for the node. This link key SHALL be randomly generated or be derived via a manufacturer specific algorithm, but it SHALL NOT be all zeros and it SHALL NOT be identical to the *LinkKey* value of the *apsDeviceKeyPairSet* entry corresponding to the joining node.
  - a. The value of the link key SHALL be stored in *bdbJoiningNodeNewTCLinkKey*
  - b. The Trust Center SHALL issue the *APSME-TRANSPORT-KEY.request* primitive to the APSME encrypted with the *LinkKey* value of the *apsDeviceKeyPairSet* entry corresponding to the joining node.
9. If, within the timeout initiated in step 1, the Trust Center receives an *APSME-VERIFY-KEY.indication* with a *SrcAddress* field equal to *bdbJoiningNodeEui64* it SHALL do the following.
  - a. It SHALL find the entry in the *apsDeviceKeyPairSet* where the *DeviceAddress* corresponds to the *bdbJoiningNodeEui64*.

- 2029           b. If the value of *bdbJoiningNodeNewTCLinkKey* is different than the  
 2030           value of the *LinkKey* of the *apsDeviceKeyPairSet* entry, the Trust  
 2031           Center:  
 2032                i. MAY set *OutgoingFrameCounter* to 0 and SHALL set  
 2033                *IncomingFrameCounter* to 0 within the *apsDeviceKeyPairSet*  
 2034                entry.  
 2035                ii. SHALL copy the *bdbJoiningNodeNewTCLinkKey* value to the  
 2036                *LinkKey* value of the *apsDeviceKeyPairSet*.  
 2037           c. It SHALL issue the APSME-CONFIRM-KEY.request primitive with  
 2038           the *DestAddress* field set to *bdbJoiningNodeEui64*.  
 2039           d. It SHALL then continue from step 12.
- 2040   10. If *bdbTrustCenterRequireKeyExchange* is equal to FALSE (the link key does  
 2041   not have to be exchanged), the Trust Center SHALL continue from step 12.
- 2042   11. The Trust Center SHALL request that the joining node leave the network. To  
 2043   do this, the Trust Center issues the APSME-REMOVE-DEVICE.request  
 2044   primitive with the *ParentAddress* parameter set to the *SrcAddress* parameter  
 2045   from the APSME-UPDATE-DEVICE.indication primitive, received in step 1,  
 2046   and the *ChildAddress* parameter set to *bdbJoiningNodeEui64*.
- 2047   12. The Trust Center SHALL do the following before terminating the procedure  
 2048   for adding a new node into the network:  
 2049        a. Expire the *bdbTrustCenterNodeJoinTimeout* timer.  
 2050        b. Set the value of the *bdbJoiningNodeNewTCLinkKey* to zero.  
 2051        c. Set the value of the *bdbJoiningNodeEui64* to zero.

### 2052   10.3.3 Behavior when a known node joins

2053   If a node that has already exchanged its Trust Center link key attempts to join an open  
 2054   Trust Center a second time, i.e. the *DeviceAddress* parameter of the APSME-  
 2055   UPDATE-DEVICE.indication primitive corresponds to an entry in  
 2056   *apsDeviceKeyPairSet* with the *KeyAttributes* field equal to VERIFIED\_KEY, the  
 2057   Trust Center SHALL allow the node to join but in a fresh state and use the initial link  
 2058   key appropriate for the node when transferring the network key. Under these  
 2059   circumstances, the Trust Center SHALL use the following steps in place of steps 4  
 2060   and 5 of the procedure given in 10.3.2:

- 2061   4. If *bdbJoinUsesInstallCodeKey* is equal to TRUE and the installation code  
 2062   derived link key is not stored, the Trust Center SHALL terminate the  
 2063   procedure for adding a new node into the network. If *bdbJoinUsesInstall-  
 2064   CodeKey* is equal to TRUE and the installation code derived link key is stored,  
 2065   the Trust Center SHALL first find the entry in *apsDeviceKeyPairSet* that  
 2066   corresponds to the joining node and then overwrite the *LinkKey* entry with the  
 2067   installation code derived link key and set the *KeyAttributes* field to  
 2068   PROVISIONAL\_KEY. The Trust Center MAY then set *OutgoingFrame-  
 2069   Counter* to 0 and SHALL set *IncomingFrameCounter* to 0.

2070 5. If *bdbJoinUsesInstallCodeKey* is equal to FALSE, the Trust Center SHALL  
2071 first find the entry in *apsDeviceKeyPairSet* that corresponds to the joining  
2072 node and then overwrite the *LinkKey* entry with the default global Trust Center  
2073 link key and set the *KeyAttributes* field to PROVISIONAL\_KEY. The Trust  
2074 Center MAY then set *OutgoingFrameCounter* to 0 and SHALL set  
2075 *IncomingFrameCounter* to 0.

## 2076 **10.4 Distributed security network behavior**

### 2077 **10.4.1 Adding a new node into the network**

2078 When a node operating on a distributed security network is accepting a new node for  
2079 joining it SHALL use the distributed security global link key (see 6.3.2) to encrypt the  
2080 network key.

2081 **11 Annex A: Recommended practices**

2082 **11.1 Recommendations for centralized commissioning**

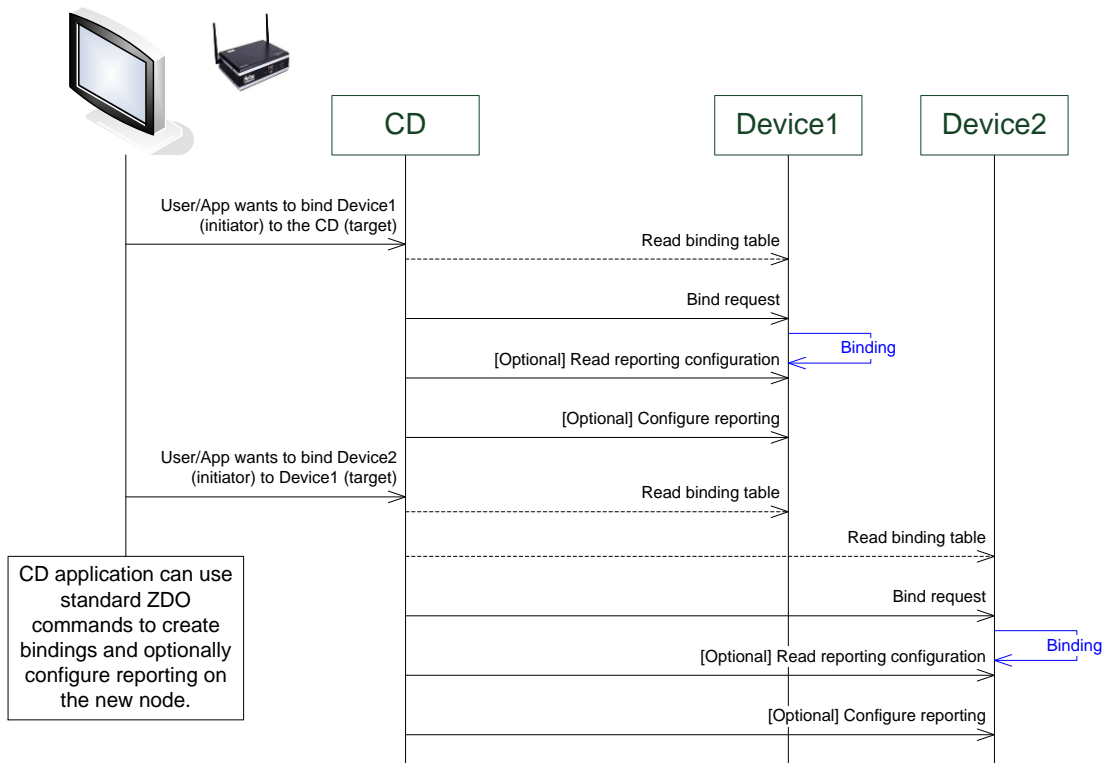
2083 **11.1.1 Centralized commissioning overview**

2084 Centralized commissioning is a method that allows a fixed or mobile node to  
 2085 commission (determine application linkages and create bindings) other nodes on the  
 2086 same network. This may also be referred to as Gateway, Tool, or S-Mode  
 2087 commissioning.

2088 This can be a node such as a gateway, a central controller or a commissioning tool  
 2089 that is typically connected to a graphical user interface. This node is able to configure  
 2090 bindings and reporting on other nodes in the network. It may also be a node that  
 2091 automatically commissions other nodes on the network from a fixed pre-loaded  
 2092 configuration.

2093 Any node in the network with this functionality is defined as a Commissioning  
 2094 Director (CD).

2095



2096  
2097

2098 **Figure 16 – Principle of centralized commissioning with a**  
 2099 **commissioning director**

2100 **11.1.2 Recommendations for device discovery**

2101 In order to commission nodes, the CD needs to discover the devices in the network.  
 2102 Recommended methods to discover all nodes in the network are listed below.

**2103 11.1.2.1 New nodes joining**

2104 A new node that joins the network is announced by a broadcast ZDO command  
2105 *Device\_annce*. A CD may then use ZDO discovery services to understand the node in  
2106 the network, binding table services to manage binding tables and, if required, the  
2107 *groups* cluster commands to manage group tables.

**2108 11.1.2.2 Nodes in existing network**

2109 When a CD joins an existing network, it needs to discover nodes already in the  
2110 network. The CD MAY initiate this process immediately on successfully joining a  
2111 network or on some user stimulus. In addition, the CD MAY periodically discover  
2112 nodes on the network in order to keep abreast of any changes.

2113 There are several ways for a CD to discover nodes in the network but it is  
2114 RECOMMENDED that the CD uses the *Mgmt\_Lqi\_req* ZDO command. The benefits  
2115 of using *Mgmt\_Lqi\_req* (instead of *IEEE\_addr\_req* or *NWK\_addr\_req*) are listed  
2116 below:

- 2117 • ZigBee logical device type information of ZigBee coordinator, ZigBee router,  
2118 ZigBee end device
- 2119 • Rx\_On\_when\_Idle information
- 2120 • Information about parent-child relationships

2121

2122 After the CD has performed device discovery, it MAY perform further commission  
2123 actions such as setting up bindings or configuring reportings.

**2124 11.1.2.3 Establishing communications with end devices**

2125 This section is a placeholder for recommendations for a CD to communicate with end  
2126 devices and will be added when the use cases are better understood.

2127