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

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

http://www.zigbee.org

1

2 This page is intentionally blank

Notice of use and disclosure

35 This page is intentionally blank

Revision history

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

39

40 This page is intentionally blank

41

Table of Contents

This page is intentionally blank

List of Figures

165

166 This page is intentionally blank

167

168 # List of Tables

177

178

179 This page is intentionally blank

180

1 Introduction

1.1 Scope

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

- The environment required for the base device
- The initialization procedures of the base device
- The commissioning procedures of the base device
- The reset procedures of the base device
- The security procedures of the base device

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

1.2 Purpose

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

1.3 Conformance levels

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

1.4 Conventions

1.4.1 Number formats

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

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

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

1.5 Conformance testing

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

1.6 Errata

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

2 References[1]

2.1 ZigBee Alliance documents

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

2.2 IEEE documents

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

2.3 IETF documents

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

[1] The version and date information in these references was correct at the time this document was released.

3 Definitions

Application cluster:

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

Application transaction:

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

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

Bind or binding (verb):

Create a binding or the act of creating a binding.

Binding (noun):

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

Centralized security network:

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

Commissioning director:

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

Device:

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

Distributed security network:

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

Dynamic device:

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

EZ-Mode:

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

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

EZ-Mode finding & binding:

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

EZ-Mode network steering:

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

Finding & binding:

See *EZ-Mode finding & binding.*

Initiator cluster:

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

Joined:

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

Network steering:

See *EZ-Mode network steering.*

Node:

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

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

Simple device:

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

Target cluster:

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

Touchlink commissioning:

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

Utility cluster:

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

ZigBee coordinator:

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

ZigBee end device:

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

ZigBee router:

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

4 Acronyms and abbreviations

AES	Advanced Encryption Standard
AIB	Application support sub-layer information base
APS	Application support sub-layer
APSME	Application support sub-layer management entity
CBKE	Certificate based key exchange
CCITT	Comité Consultatif International Téléphonique et Télégraphique
CD	Commissioning director
CRC	Cyclic redundancy check
EP	Endpoint
EUI	Extended unique identifier
ID	Identifier
IEEE	Institute of Electrical and Electronic Engineers
LQI	Link quality indication
MAC	Medium access control
MMO	Matyas-Meyer-Oseas
NLME	Network layer management entity
NVRAM	Non-volatile random access memory
NWK	Network
OTA	Over the air
PAN	Personal area network
PHY	Physical
TC	Trust Center
WPAN	Wireless personal area network
ZC	ZigBee coordinator
ZCL	ZigBee cluster library
ZDO	ZigBee device objects
ZED	ZigBee end device
ZR	ZigBee router

5 Environment variables

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

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

5.1 Constants used by all nodes

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

Table 1 – Constants used by all nodes

Constant	Value
bdbcMaxSameNetworkRetryAttempts	10
bdbcMinCommissioningTime	180s (0xb4)
bdbcRecSameNetworkRetryAttempts	3
bdbcTCLinkKeyExchangeTimeout	5s

5.1.1 *bdbcMaxSameNetworkRetryAttempts* constant

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

This constant is used by each node.

See also *bdbcRecSameNetworkRetryAttempts*.

5.1.2 *bdbcMinCommissioningTime* constant

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

This constant is used by each node.

5.1.3 *bdbcRecSameNetworkRetryAttempts* constant

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

This constant is used by each node.

See also *bdbcMaxSameNetworkRetryAttempts*.

5.1.4 *bdbcTCLinkKeyExchangeTimeout* constant

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

This constant is used by each node.

5.2 Constants used by nodes supporting touchlink

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

Table 2 – Constants used by nodes supporting touchlink

Constant	Value
bdbcTLInterPANTransIdLifetime	8s
bdbcTLMinStartupDelayTime	2s
bdbcTLPrimaryChannelSet	0x02108800
bdbcTLRxWindowDuration	5s
bdbcTLScanTimeBaseDuration	0.25s
bdbcTLSecondaryChannelSet	0x07fff800 XOR *bdbcTLPrimaryChannelSet*

5.2.1 *bdbcTLInterPANTransIdLifetime* constant

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

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

5.2.2 *bdbcTLMinStartupDelayTime* constant

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

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

5.2.3 *bdbcTLPrimaryChannelSet* constant

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

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

5.2.4 *bdbcTLRxWindowDuration* constant

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

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

5.2.5 *bdbcTLScanTimeBaseDuration* constant

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

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

5.2.6 *bdbcTLSecondaryChannelSet* constant

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

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

5.3 Attributes

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

Table 3 – Attributes used in the base device behavior

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

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

***bdbCommissioning-Mode* 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[2]
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

[2] 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**

Value of the *bdbCommissioningStatus* attribute	Description
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 *identify query* 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 *scan request* 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.

This attribute is used by ZigBee coordinator nodes.

5.3.6 *bdbJoinUsesInstallCodeKey* attribute

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

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

This attribute is used by ZigBee coordinator nodes.

5.3.7 *bdbNodeCommissioningCapability* attribute

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

This attribute is used by all nodes.

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

***bdbCommissioning-Capability* 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.

5.3.9 *bdbNodeJoinLinkKeyType* attribute

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

Table 7 – Values of the *bdbNodeJoinLinkKeyType* attribute

Value of the *bdbNodeJoinLinkKeyType* 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

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

5.3.10 *bdbPrimaryChannelSet* attribute

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

This attribute is used by all nodes.

5.3.11 *bdbScanDuration* attribute

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

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

This attribute is used by all nodes.

5.3.12 *bdbSecondaryChannelSet* attribute

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

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 *bdbTCLinkKeyExchangeMethod* 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.

6 General requirements

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

6.1 ZigBee logical device types

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

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

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

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

6.2 Network security models

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

6.3 Link keys

Each node SHALL contain the following link keys:

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

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

4. The touchlink preconfigured link key

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

6.3.1 Default global Trust Center link key

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

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

6.3.2 Distributed security global link key

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

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

6.3.3 Install code derived preconfigured link key

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

6.3.4 Touchlink preconfigured link key

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

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

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

6.4 Use of install codes

All nodes SHALL support install codes.

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

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

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

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

6.5 Commissioning

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

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

This specification specifies the procedures for the following commissioning mechanisms:

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

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

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

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

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

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

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

6.6 Minimum requirements for all devices

All nodes SHALL support the following requirements:

- A node SHALL process the ZDO discovery service commands: *Active_EP_req*, *Node_Desc_req*, *Simple_Desc_req*, *IEEE_addr_req*, *NWK_addr_req* and *Match_Desc_req* and respond with the *Active_EP_rsp*,

Node_Desc_rsp, *Simple_Desc_rsp*, *IEEE_addr_rsp*, *NWK_addr_rsp* and *Match_Desc_rsp* commands, respectively.

- A node SHALL process the ZDO node manager service commands *Mgmt_Bind_req* and *Mgmt_Lqi_req* and respond with the *Mgmt_Bind_rsp* and *Mgmt_Lqi_rsp* commands, respectively.
- A node SHALL process the ZDO binding table service commands *Bind_req* and *Unbind_req* and respond with the *Bind_rsp* and *Unbind_rsp* commands, respectively.
- A node SHALL process the ZDO network manager service command *Mgmt_Leave_req* and respond with the *Mgmt_Leave_rsp* command.
- A node SHALL be able to handle receiving at least one *Identify* cluster, *Identify Query Response* command frame after broadcasting an *Identify Query* command frame during finding & binding. If the node is able to handle receiving more than one *Identify Query Response* command frames, how this is handled is implementation specific.
- A node that supports finding & binding as an initiator SHALL implement a binding table whose number of available entries is greater than or equal to the sum of the cluster instances, supported on each device of the node, that are initiators of application transactions. Bindings are configured in the binding table during finding & binding, touchlink or centralized commissioning. Regardless of the commissioning mechanism used to generate the bindings, the binding table SHALL be consistent such that its contents can be retrieved using the *Mgmt_Bind_req* command.
- A node SHALL have a default report configuration (see sub-clause 6.7) for every implemented attribute that is specified as mandatory and reportable.
- A node that can be a target of an application transaction SHALL support group addressing and at least 8 memberships in the group table.

6.7 Default reporting configuration

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

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

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

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

6.8 MAC data polling

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

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

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

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

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

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

6.9 ZigBee persistent data

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

- *bdbNodeIsOnANetwork* attribute.

7 Initialization

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

7.1 Initialization procedure

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

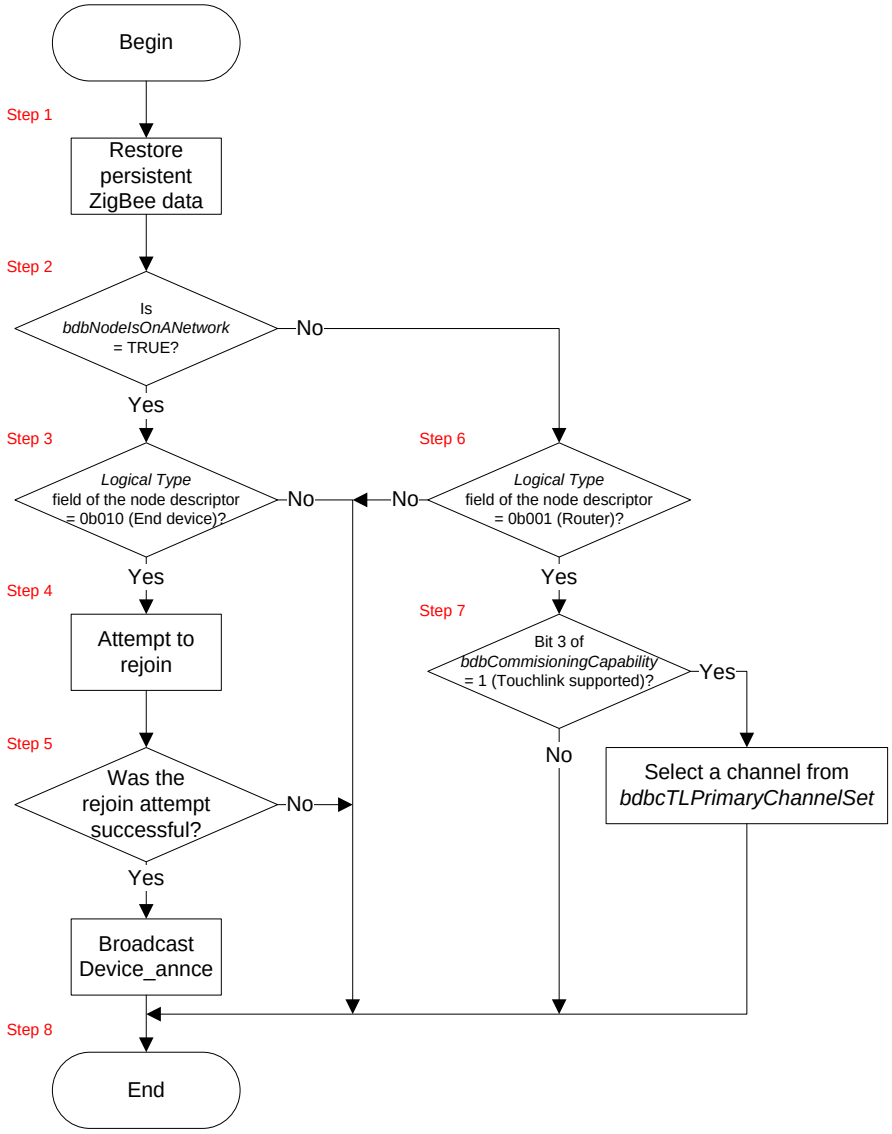


Figure 1 – Initialization procedure

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

8 Commissioning

Commissioning MAY be invoked when a node is not on a network, on a network but not bound to another device or on a network and bound to another device. Commissioning MAY be triggered by a user interaction, via some over the air mechanism (such as that defined in the *Identify* cluster) or invoked directly by application software (such as automatically after initialization). The commissioning procedures specified in this section define the steps and states when commissioning is invoked.

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

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

8.1 Top level commissioning procedure

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

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

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

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

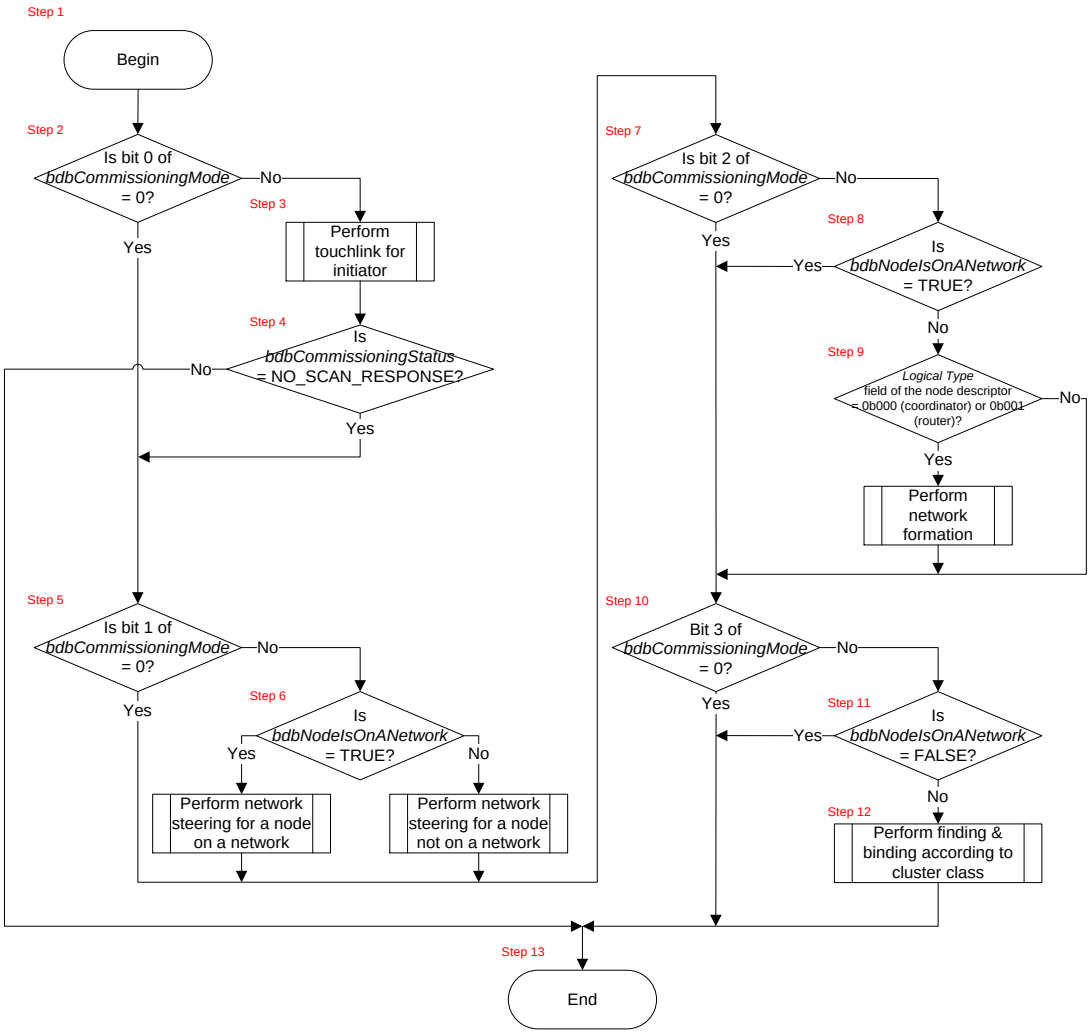


Figure 2 – Top level commissioning procedure

1. On receipt of an application stimulus for commissioning, the device first sets *bdbCommissioningStatus* to SUCCESS and then determines the required commissioning steps by inspecting *bdbCommissioningMode*.
2. If bit 0 of *bdbCommissioningMode* is equal to 0 (i.e. touchlink is not required), the device SHALL continue from step 5.
3. The node SHALL follow the touchlink procedure as an initiator (see sub-clause 8.7).
4. If *bdbCommissioningStatus* is not equal to NO_SCAN_RESPONSE (i.e. there was a response to the touchlink scan request from the initiator, indicating a successful touchlink), the device SHALL continue from step 13.
5. If bit 1 of *bdbCommissioningMode* is equal to 0 (i.e. network steering is not required), the device SHALL continue from step 7.

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

8.2 Network steering procedure for a node on a network

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

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

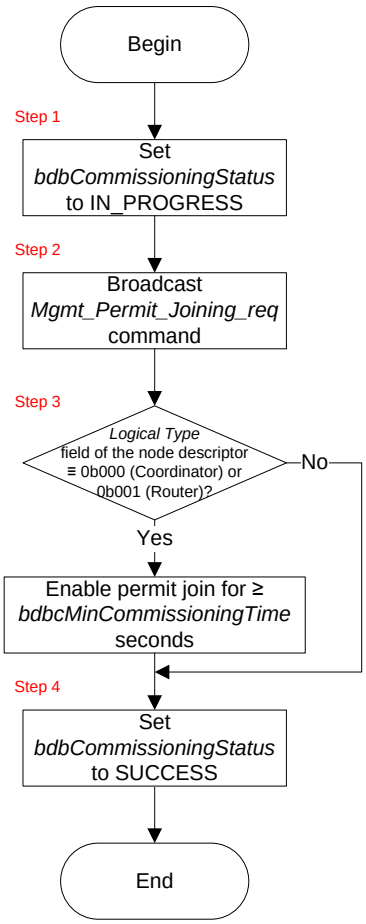


Figure 3 – Network steering procedure for a node on a network

1. The node first sets *bdbCommissioningStatus* to IN_PROGRESS.
2. The node SHALL broadcast the *Mgmt_Permit_Joining_req* ZDO command with the *PermitDuration* field set to at least *bdbcMinCommissioningTime* and the *TC_Significance* field set to 0x01.
3. If the *logical type* field of the node descriptor for the node is equal to 0b000 (ZigBee coordinator) or 0b001 (ZigBee router), the node issues the *NLME-PERMIT-JOINING.request* primitive with the *PermitDuration* parameter set to at least *bdbcMinCommissioningTime*. On receipt of the *NLME-PERMIT-JOINING.confirm* primitive from the NWK layer, the node is notified of the status of the request to activate permit joining.
4. The node then sets *bdbCommissioningStatus* to SUCCESS and it SHALL terminate the network steering procedure for a node on a network.

8.3 Network steering procedure for a node not on a network

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

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

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

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

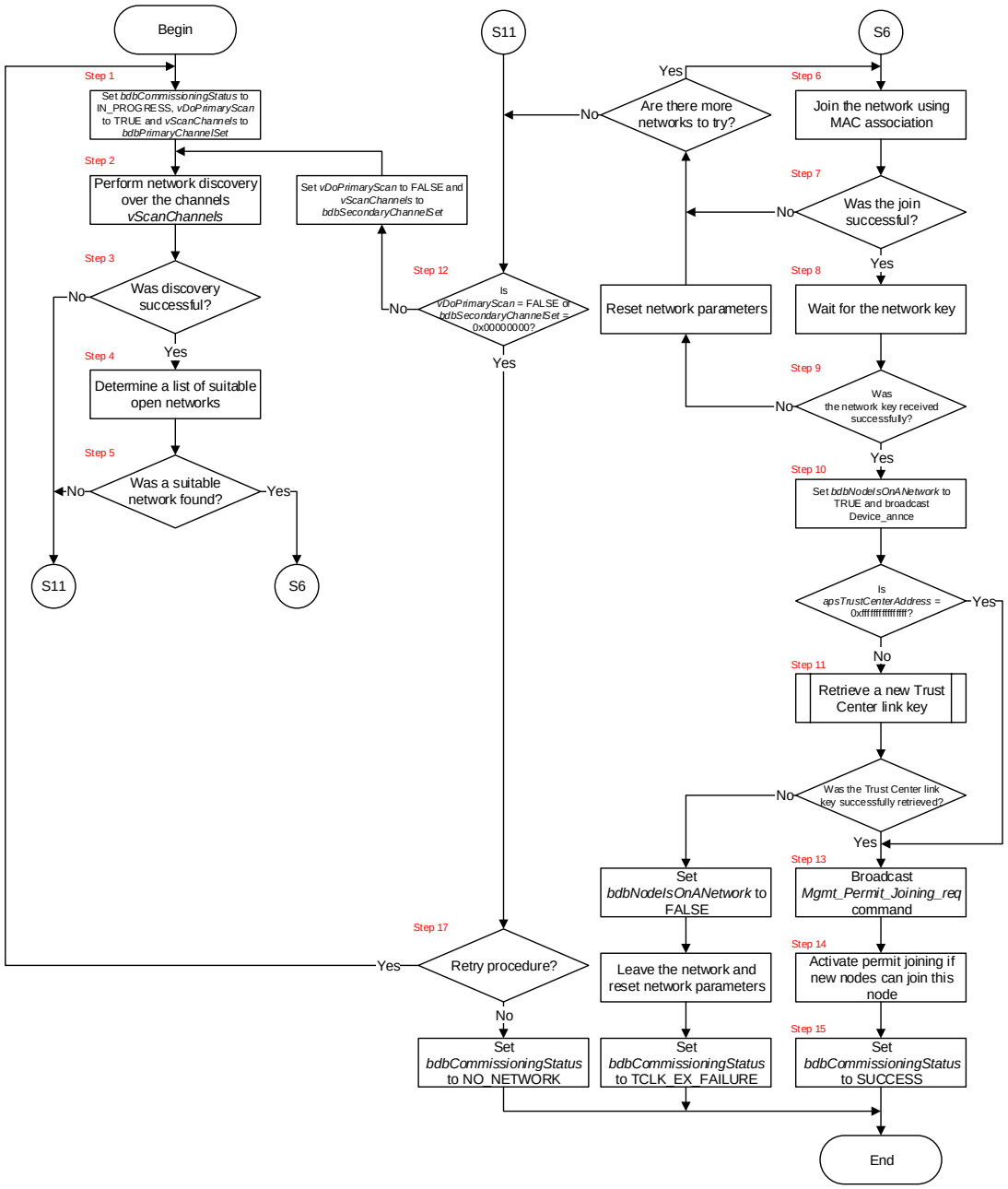


Figure 4 – Network steering procedure for a node not on a network

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

issues the *NLME-NETWORK-DISCOVERY.request* primitive with the *ScanChannels* parameter set to *vScanChannels* and the *ScanDuration* parameter set to *bdbScanDuration*. On receipt of the *NLME-NETWORK-DISCOVERY.confirm* primitive from the NWK layer, the node is notified of the status of the request to discover networks.

3. If the *Status* parameter from the *NLME-NETWORK-DISCOVERY.confirm* primitive is not equal to *SUCCESS*, indicating that the channel scan was not successful, the node SHALL continue from step 12.
4. The node SHALL determine whether any suitable networks with a permit joining flag set to TRUE were found by analyzing the *NetworkCount* and *NetworkDescriptor* parameters. The decision regarding what constitutes a *suitable* network is application specific.
5. If a suitable network is not found on the channel scan, the node SHALL continue from step 12.
6. The node SHALL attempt to join the network found using MAC association. To do this, the node issues the *NLME-JOIN.request* primitive with the *ExtendedPANId* parameter set to the extended PAN identifier of the selected network, the *RejoinNetwork* parameter set to 0x00, the *ScanChannels* parameter set to 0x00000000, the *ScanDuration* parameter set to 0x00, the *CapabilityInformation* set appropriately for the node and the *SecurityEnable* parameter set to FALSE. On receipt of the *NLME-JOIN.confirm* primitive from the NWK layer, the node is notified of the status of the request to join the network using MAC association.
7. If the *Status* parameter from the *NLME-JOIN.confirm* primitive is not equal to *SUCCESS*, indicating that the join was not successful, the node SHALL try to join the next suitable network from step 6. Note that it is permissible to try to join the same network again, but this SHALL NOT be attempted more than *bdbcMaxSameNetworkRetryAttempts* times in succession (*bdbcRecSame-NetworkRetryAttempts* times in succession is RECOMMENDED). If there are no further suitable networks to join the node SHALL continue from step 12.
8. If the *Status* parameter from the *NLME-JOIN.confirm* primitive is equal to *SUCCESS*, indicating that the join was successful, the node SHALL wait for at most *apsSecurityTimeOutPeriod* milliseconds to be authenticated and receive the network key from its parent. Note that the network key may be tunneled from the Trust Center in a centralized security network, encrypted using the default global Trust Center link key or via an install code derived preconfigured link key, or directly from its parent in a distributed security network, encrypted using the distributed security global link key. The node SHALL set *bdbNodeJoinLinkKeyType* accordingly to indicate the type of link key used to decrypt the received network key.
9. If the node does not receive the network key from its parent within *apsSecurityTimeOutPeriod* milliseconds, the network key is received within

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

10. The node sets *bdbNodeIsOnANetwork* to TRUE and then broadcasts a Device_annce ZDO command. If *apsTrustCenterAddress* is equal to 0xffffffffffffffff, the node SHALL continue from step 13.
11. The node SHALL perform the procedure for retrieving a new Trust Center link key (see sub-clause 10.2.5). If the procedure is successful, the node SHALL continue from step 13. If the procedure is not successful, the node SHALL perform a leave request on its old network and resets its network parameters. The node then sets *bdbNodeIsOnANetwork* to FALSE and sets *bdbCommissioningStatus* to TCLK_EX_FAILURE. To perform a leave request, the node issues the *NLME-LEAVE.request* primitive to the NWK layer with the *DeviceAddress* parameter set to NULL, the *RemoveChildren* parameter set to FALSE and the *Rejoin* parameter set to FALSE. On receipt of the *NLME-LEAVE.confirm* primitive, the node is notified of the status of the request to leave the network. The node SHALL then terminate the network steering procedure for a node not on a network.
12. If *vDoPrimaryScan* is equal to FALSE or *bdbSecondaryChannelSet* is equal to 0x00000000, the node SHALL continue from step 16. If *bdbSecondaryChannelSet* is not equal to 0x00000000, the node SHALL set *vDoPrimaryScan* to FALSE, set *vScanChannels* to *bdbSecondaryChannelSet* and continue from step 2.
13. The node SHALL broadcast the *Mgmt_Permit_Joining_req* ZDO command with the *PermitDuration* field set to at least *bdbcMinCommissioningTime* and the *TC_Significance* field set to 0x01. Note that this will cause nodes receiving this command to reset the timer, during which their permit joining flag is activated, thus extending the time for further new nodes to join.
14. If the node is able to allow new nodes to join, it SHALL activate its permit joining flag. To do this, the node issues the *NLME-PERMIT-JOINING.request* primitive with the *PermitDuration* parameter set to at least *bdbcMinCommissioningTime*. On receipt of the *NLME-PERMIT-JOINING.confirm* primitive from the NWK layer, the node is notified of the status of the request to activate permit joining.
15. The node then sets *bdbCommissioningStatus* to SUCCESS. If the node supports touchlink, it sets the values of the *aplFreeNwkAddrRangeBegin*, *aplFreeNwkAddrRangeEnd*, *aplFreeGroupID-RangeBegin* and

aplFreeGroupIDRangeEnd attributes all to 0x0000 (indicating the node having joined the network using MAC association). The node SHALL then terminate the network steering procedure for a node not on a network.

16. The node MAY retry using some manufacturer specific procedure OR set *bdbCommissioningStatus* to NO_NETWORK and then it SHALL terminate the network steering procedure for a node not on a network. If a manufacturer specific procedure is attempted, the *bdbCommissioningStatus* and *bdbNodeIsOnANetwork* attributes are updated accordingly on its termination so that the commissioning procedure is consistent.

8.4 Network formation procedure

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

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

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

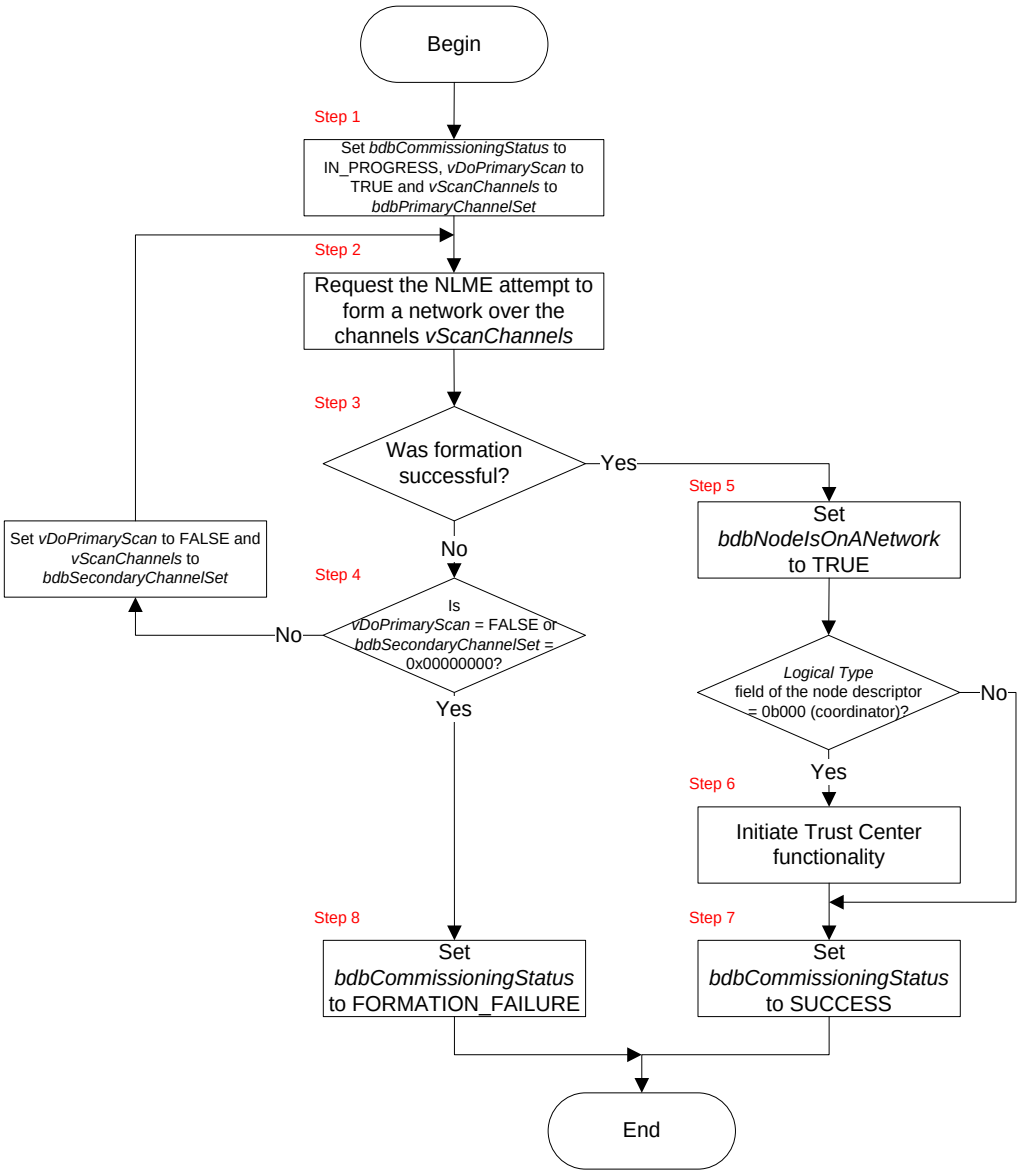


Figure 5 – Network formation procedure

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.

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

8.5 Finding & binding procedure for a target endpoint

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

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

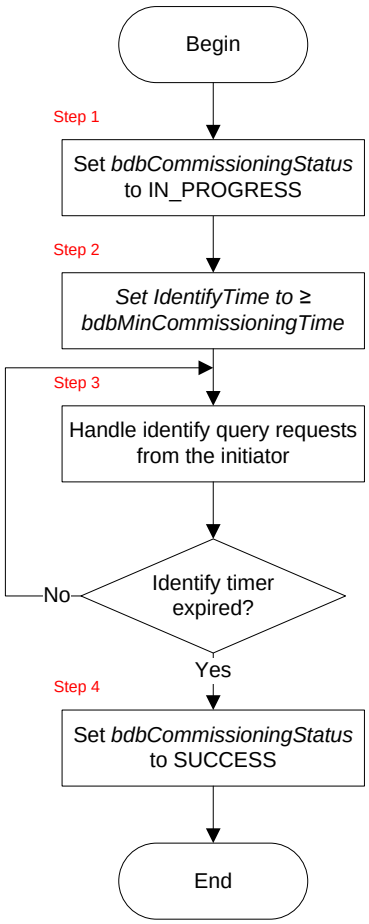


Figure 6 – Finding & binding procedure for a target endpoint

1. The target device first sets *bdbCommissioningStatus* to IN_PROGRESS.
2. The target device SHALL set the *Identify* cluster, *IdentifyTime* attribute to at least *bdbcMinCommissioningTime*. The target device MAY also set the *Identify* cluster, *IdentifyTime* attribute to at least *bdbcMinCommissioningTime* on any other identifying endpoints.
3. During the *IdentifyTime*, the target device SHALL respond to the identify queries. They may be followed by other finding & binding commands; those SHALL be handled irrespective of the identify status.
4. When the decrementing *IdentifyTime* attribute reaches zero, the target device sets *bdbCommissioningStatus* to SUCCESS and it SHALL terminate the finding & binding procedure for a target endpoint.

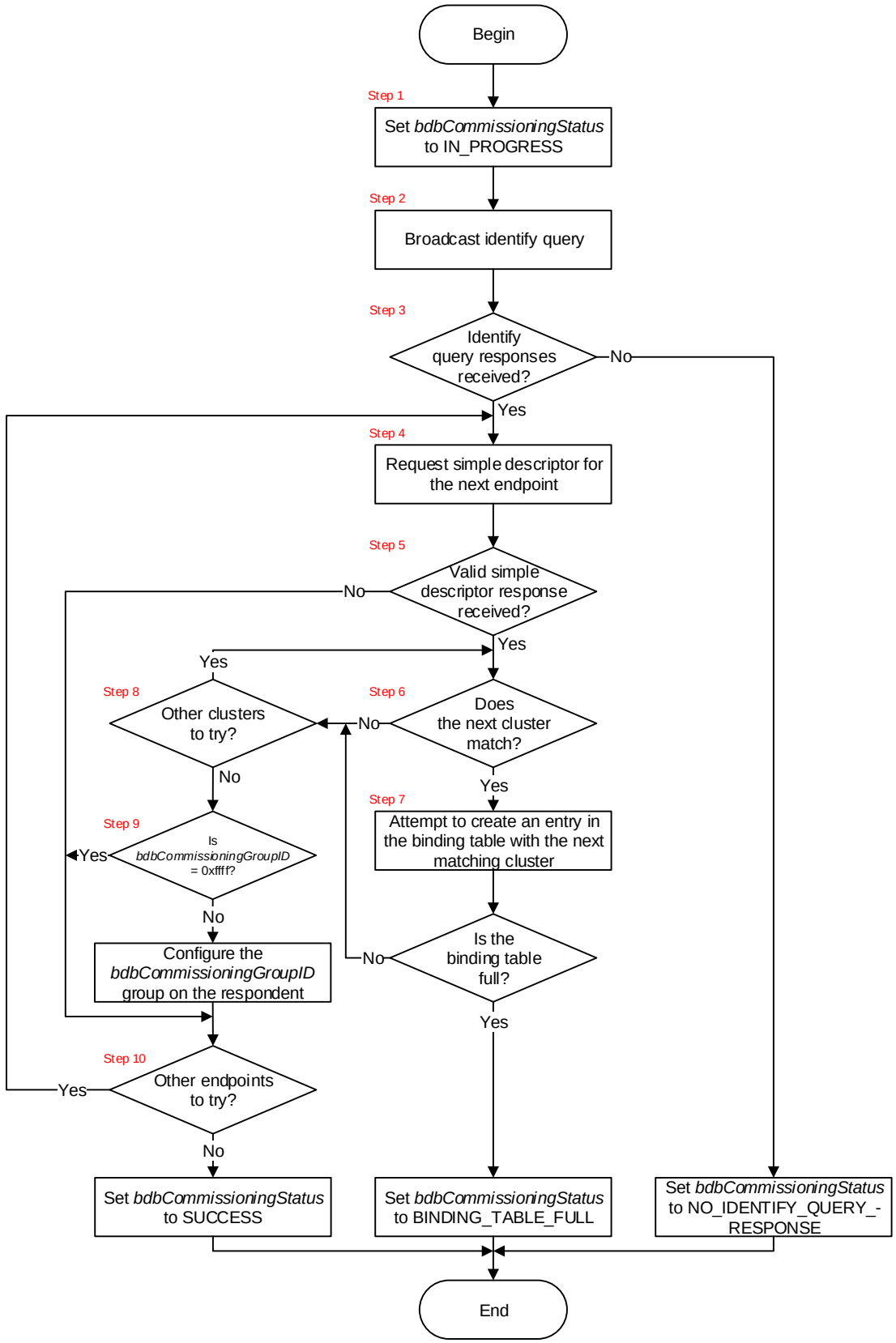
8.6 Finding & binding procedure for an initiator endpoint

This section defines the finding & binding procedure for an initiator endpoint. In this procedure, the initiator endpoint first searches for identifying target endpoints and if 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

found, it creates a corresponding entry in its binding table. If a group binding is requested, the initiator endpoint configures group membership of the target endpoint.

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**

1. The initiator device first sets *bdbCommissioningStatus* to IN_PROGRESS.
2. The initiator device SHALL broadcast the *Identify* cluster, *Identify Query* command from the initiator endpoint to all nodes (i.e., using the broadcast address 0xffff). The initiator device MAY broadcast this command one or more times.
3. If no *Identify* cluster, *Identify Query Response* commands are received, the initiator device sets *bdbCommissioningStatus* to NO_IDENTIFY_QUERY_RESPONSE and it SHALL terminate the finding & binding procedure for an initiator endpoint. If at least one *Identify* cluster, *Identify Query Response* command is received, the initiator device SHALL note the NWK address, contained in the *source address* field of the NWK header, and the endpoint, contained in the *source endpoint* field of the APS header, of each incoming frame from the target devices that responded; such a device is referred to as a "respondent".
4. The initiator device SHALL obtain the simple descriptor for the next response endpoint from a respondent. To do this, the initiator device SHALL unicast the *Simple_Desc_req* ZDO command to the respondent with the *NWKAddrOfInterest* field set to the NWK address of the respondent and the *EndPoint* field set to the identifier of the endpoint being addressed (found from the APS header of the respondent's *Identify* cluster, *Identify Query Response* command).
5. If a *Simple_Desc_rsp* ZDO command is not received from the respondent or a *Simple_Desc_rsp* ZDO command is received with the *Status* field not equal to SUCCESS, the initiator device SHALL continue from step 10.
6. The initiator SHALL check the next application target cluster listed in the *Application Input Cluster List* or *Application Output Cluster List* fields of the simple descriptor of the respondent and if the initiator device does not support the corresponding client/server cluster, the initiator device SHALL continue from step 8.
7. If the initiator is a simple device, it SHALL create a binding table entry for that cluster. Conversely, if the initiator is not a simple device, it MAY create a binding table entry for that cluster. If a unicast binding table entry is to be created (i.e., if *bdbCommissioningGroupId* is equal to 0xffff) and the IEEE address of the respondent is not known, the initiator SHALL obtain it using the *IEEE_addr_req* ZDO command before creating a binding. To create a binding table entry, the initiator device issues the *APSME-BIND.request* primitive with the *SrcAddr* parameter set to the IEEE address of the initiator device (*aExtendedAddress*), the *SrcEndpoint* parameter set to the identifier of the initiator endpoint and the *ClusterId* parameter set to the identifier of the matching cluster. The *DstAddrMode* and *DstAddr* parameters SHALL be set to 0x01 and *bdbCommissioningGroupId*, respectively, (if

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

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

8.7 Touchlink procedure for an initiator

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

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

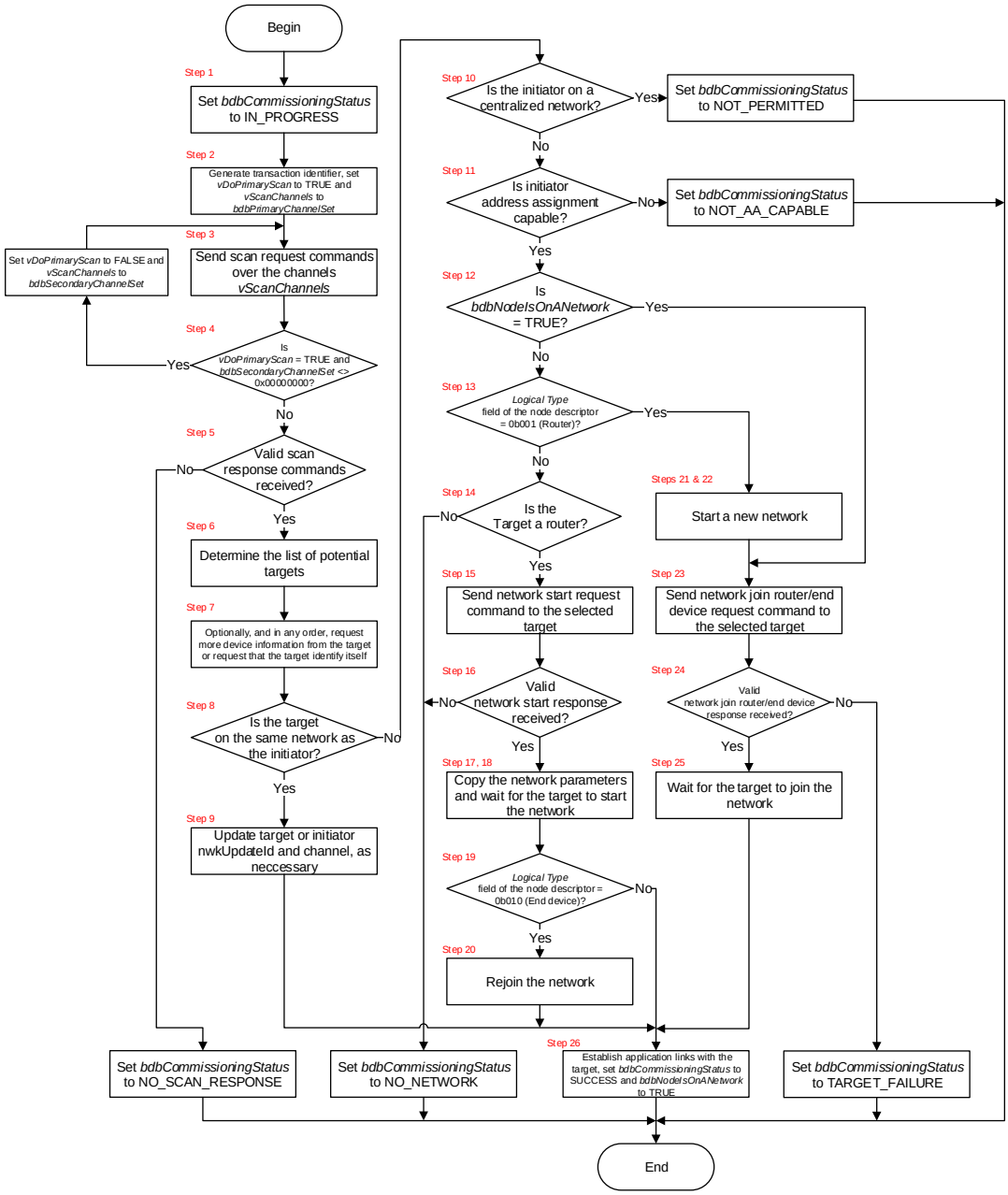
The touchlink procedure for an initiator can perform a “normal” channel scan or an “extended” channel scan; the latter is used if a reset to factory new is required (see sub-clause 9.2) or if the target could be operating on a channel other than those defined in *bdbcTLPrimaryChannelSet*. For a normal 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 **Figure 8 – Touchlink procedure for an initiator**

1216

1217 1. The initiator first sets *bdbCommissioningStatus* to IN_PROGRESS.

2. The initiator SHALL generate a 32-bit transaction identifier to use in the *inter-PAN transaction identifier* fields of all commands used in the touchlink procedure. The transaction identifier SHALL be random, non-zero and non-sequential. The initiator then sets *vDoPrimaryScan* to TRUE, *vScanChannels* set to *bdbPrimaryChannelSet* and *vIsFirstChannel* set to TRUE. If *bdbPrimaryChannelSet* is equal to 0x00000000, the node SHALL continue from step 4.
3. The initiator SHALL perform touchlink device discovery. If *vIsFirstChannel* is equal to TRUE, the initiator SHALL set *vIsFirstChannel* to FALSE, switch to the first channel defined by *vScanChannels* and broadcast five consecutive *touchlink commissioning* cluster *scan request* inter-PAN command frames. The initiator SHALL then switch to each of the remaining channels specified in *vScanChannels* in turn and broadcast a single *scan request* inter-PAN command frame on each channel. Each *scan request* inter-PAN command frames SHALL be broadcast with appropriate values for the *ZigBee information* and *touchlink information* fields and with a nominal output power of 0dBm. After each transmission, the initiator SHALL wait *bdbcTLScanTimeBaseDuration* seconds to receive any responses. If, during its scan, an initiator with the *bdbNodeIsOnANetwork* attribute equal to FALSE receives another *scan request* inter-PAN command frame with the *factory new* sub-field of the *touchlink information* field equal to 1, it SHALL be ignored. Conversely, if, during its scan, an initiator with the *bdbNodeIsOnANetwork* attribute equal to FALSE receives another *scan request* inter-PAN command frame with the *factory new* sub-field of the *touchlink information* field equal to 0, it MAY stop sending its own *scan request* inter-PAN command frames and assume the role of a target (see sub-clause 8.8), responding with a *touchlink commissioning* cluster *scan response* inter-PAN command frame and remaining on the same channel for further touchlink command frames. Touchlink device discovery MAY be aborted at any time. Since no node parameters such as network settings are altered, this step is non-intrusive for the nodes involved.
4. If *vDoPrimaryScan* is equal to TRUE and *bdbSecondaryChannelSet* is not equal to 0x00000000, the node sets *vDoPrimaryScan* to FALSE, set *vScanChannels* to *bdbSecondaryChannelSet* and it SHALL continue from step 3.
5. If no *touchlink commissioning* cluster *scan response* inter-PAN command frames are received or no *touchlink commissioning* cluster *scan response* inter-PAN command frames are received with the *inter-PAN transaction identifier* field equal to that used by the initiator in its *scan request* command frame, the node sets *bdbCommissioningStatus* to

NO_SCAN_RESPONSE and it SHALL terminate the touchlink procedure for an initiator.

6. Touchlink device discovery can result in more than one *touchlink commissioning* cluster *scan response* inter-PAN command frames giving a list of potential targets from which the application, via some product specific means, selects one target for further processing. If the *touchlink priority request* bit of the *touchlink information* field of the *touchlink commissioning* cluster *scan response* command frame is equal to 1, the initiator MAY consider giving priority processing to those nodes.
7. In any order, the initiator MAY request more device information from the target, if necessary, or request the selected target to identify itself in order to support a user confirmation. To request more device information from the target, the initiator SHALL generate and transmit a *touchlink commissioning* cluster *device information request* inter-PAN command frame to the appropriate discovered target and wait for a corresponding *touchlink commissioning* cluster *device information response* inter-PAN command frame (note that this is not necessary if a target has only one sub-device since its information is entirely contained in the *scan response* command frame). To request the target identify itself, the initiator SHALL generate and transmit a *touchlink commissioning* cluster *identify request* inter-PAN command frame to the appropriate discovered target. The initiator MAY send further *identify request* inter-PAN command frames to the selected target, for example, to stop the identify operation, provided it can do so within *bdbcTLInterPANTransIdLifetime* seconds of the start of the touchlink transaction. If this is not possible, a new touchlink device discovery operation SHALL be performed.
8. If the *extended PAN identifier* field of the *scan response* command frame is not equal to *nwkExtendedPANID* (i.e., the target is not on the same network as the initiator), the initiator SHALL continue from step 10.
9. If the *network update identifier* field of the *scan response* command frame is lower than *nwkUpdateId* (i.e., the target has missed a channel change), the initiator SHALL generate and transmit a *touchlink commissioning* cluster *network update request* command frame to the target with the *network update identifier* field set to *nwkUpdateId* and the *logical channel* field set to the current operating channel of the initiator. If the *network update identifier* field of the *scan response* command frame is higher than *nwkUpdateId* (i.e., the initiator has missed a channel change), the initiator SHALL set *nwkUpdateId* and its current operating channel to the values of the *network update identifier* and *logical channel* fields, respectively, from the *scan response* command frame. The initiator SHALL continue from step 26.

10. If the value of *apsTrustCenterAddress* is not equal to 0xffffffffffffffff (i.e., the initiator is on a centralized security network), the initiator sets *bdbCommissioningStatus* to NOT_PERMITTED and it SHALL terminate the touchlink procedure for an initiator.
11. If the initiator is not touchlink address assignment capable, it sets *bdbCommissioningStatus* to NOT_AA_CAPABLE and it SHALL terminate the touchlink procedure for an initiator.
12. If *bdbNodeIsOnANetwork* is equal to TRUE, the initiator SHALL continue from step 23.
13. If the *logical type* field of the node descriptor for the initiator is equal to 0b001 (ZigBee router), the initiator SHALL continue from step 21.
14. If the selected target is not a ZigBee router, the initiator sets *bdbCommissioningStatus* to NO_NETWORK and it SHALL terminate the touchlink procedure for an initiator.
15. The initiator SHALL generate and unicast a *touchlink commissioning* cluster *network start request* inter-PAN command frame to the selected target. The initiator SHALL set the *logical channel* field either to zero (indicating that the target should choose the channel) or to a channel from *bdbcTLPrimaryChannelSet* if a specific primary channel is preferred. The initiator SHALL set both the *extended PAN identifier* and *PAN identifier* fields to zero. The initiator SHALL also set the *initiator IEEE address* and *initiator network address* fields to its IEEE address and the network address it will use on the new network, respectively. All other fields SHALL be specified according to sub-clause 8.7.1.
16. The initiator SHALL then enable its receiver and wait for at most *bdbcRxWindowDuration* seconds or until a corresponding *network start response* inter-PAN command frame is received from the intended target with the same *inter-PAN transaction identifier* field matching that used by the initiator in its *scan request* command frame. If a corresponding *network start response* inter-PAN command frame is not received within *bdbcRxWindowDuration* seconds or if a corresponding *network start response* inter-PAN command frame is received within *bdbcRxWindowDuration* seconds but with a non-zero value in the *Status* parameter, the initiator sets *bdbCommissioningStatus* to NO_NETWORK and it SHALL terminate the touchlink procedure for an initiator.
17. On receipt of a *network start response* inter-PAN command frame with the *Status* parameter set to SUCCESS, the initiator SHALL copy these parameters to its network information base. The initiator SHALL determine whether an entry exists in *apsDeviceKeyPairSet* with a *DeviceAddress* field which corresponds to 0xffffffffffffffff. If such an entry does not exist, the initiator SHALL create a new entry with the *DeviceAddress* field set to 0xffffffffffffffff, the *apsLinkKeyType* field set to

0x01, the *LinkKey* field set to the distributed security global link key and both the *OutgoingFrameCounter* and *IncomingFrameCounter* fields set to 0.

18. The initiator SHALL then wait at least *bdbcTLMinStartupDelayTime* seconds to allow the target to start the network.
19. If the *logical type* field of the node descriptor for the initiator is not equal to 0b010 (ZigBee end device) or a *network start request* inter-PAN command frame was not sent, the initiator SHALL continue from step 26.
20. The initiator SHALL perform a network rejoin request. To do this, the initiator issues the *NLME-JOIN.request* primitive with the *ExtendedPANId* parameter set to the extended PAN identifier of the selected network, the *RejoinNetwork* parameter set to 0x02 (the node is joining the network using the NWK rejoining procedure), the *ScanChannels* parameter set to 0x00000000, the *ScanDuration* parameter set to 0x00, the *CapabilityInformation* set appropriately for the node and the *SecurityEnable* parameter set to TRUE. On receipt of the *NLME-JOIN.confirm* primitive from the NWK layer, the initiator is notified of the status of the request for a network rejoin. The initiator SHALL then continue from step 26.
21. The initiator SHALL perform a network discovery to establish the network parameters. To do this, the initiator issues the *NLME-NETWORK-DISCOVERY.request* primitive to the NWK layer, with the *ScanChannels* parameter set to *bdbcTLPrimaryChannelSet* and the *ScanDuration* parameter set to *bdbScanDuration*. On receipt of the *NLME-NETWORK-DISCOVERY.confirm* primitive from the NWK layer, the initiator is notified of the results. Based on these results, the initiator SHALL select suitable values for the logical channel, PAN identifier and extended PAN identifier for the network.
22. The initiator SHALL then copy the new network parameters to its network information base and start operating on the new network. To do this, the initiator issues the *NLME-START-ROUTER.request* primitive to the NWK layer with the *BeaconOrder* parameter set to 0x0f, the *SuperframeOrder* set to 0x00 and the *BatteryLifeExtension* parameter set to FALSE. On receipt of the *NLME-START-ROUTER.confirm* primitive, the initiator is notified of the status of the request to start.
23. The initiator SHALL generate and unicast a *touchlink commissioning* cluster *network join router request* or *network join end device* inter-PAN command frame to the selected target, depending on whether the target is a ZigBee router or a ZigBee end device, respectively, with the *extended PAN identifier*, *network update identifier*, *logical channel* and *PAN identifier* fields set to the corresponding network parameter values as used by the

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

24. The initiator SHALL then enable its receiver and wait for at most *bdbcRxWindowDuration* seconds or until a corresponding response inter-PAN command frame is received from the intended target with the same *inter-PAN transaction identifier* field matching that used by the initiator in its *scan request* command frame. The corresponding response to a *network join router request* and a *network join end device request* command frame is a *touchlink commissioning* cluster *network join router response* and *network join end device response* command frame, respectively. If a corresponding response inter-PAN command frame is not received within *bdbcRxWindowDuration* seconds or if a corresponding response inter-PAN command frame is received within *bdbcRxWindowDuration* seconds but with a non-zero value in the *Status* parameter, the initiator sets *bdbCommissioningStatus* to TARGET_FAILURE and it SHALL terminate the touchlink procedure for an initiator.
25. The initiator SHALL then wait at least *bdbcTLMinStartupDelayTime* seconds to allow the target to start the network or to start operating on the network correctly.
26. If the initiator is a simple device, it SHALL establish binding links in the binding table to the target. Conversely, if the initiator is not a simple device, it MAY establish binding links in the binding table to the target. If binding links are to be established, the initiator SHALL then, based on the endpoint and device identifier information received in the *scan response* and/or *device information response* inter-PAN command frames, establish binding links in the binding table for matching client/server clusters on the initiator and the corresponding server/client clusters on the target. The initiator sets *bdbCommissioningStatus* to SUCCESS, sets *bdbNodeIsOnANetwork* to TRUE and it SHALL terminate the touchlink procedure for an initiator.

8.7.1 General field settings for network start/join commands

8.7.1.1 Inter-PAN transaction identifier field

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

8.7.1.2 Key index and encrypted network key fields

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

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

8.7.1.3 Network address field

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

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

8.7.1.4 Group identifiers begin/end fields

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

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

8.7.1.5 Free network/group address range begin/end fields

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

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

8.8 Touchlink procedure for a target

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

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

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

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

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

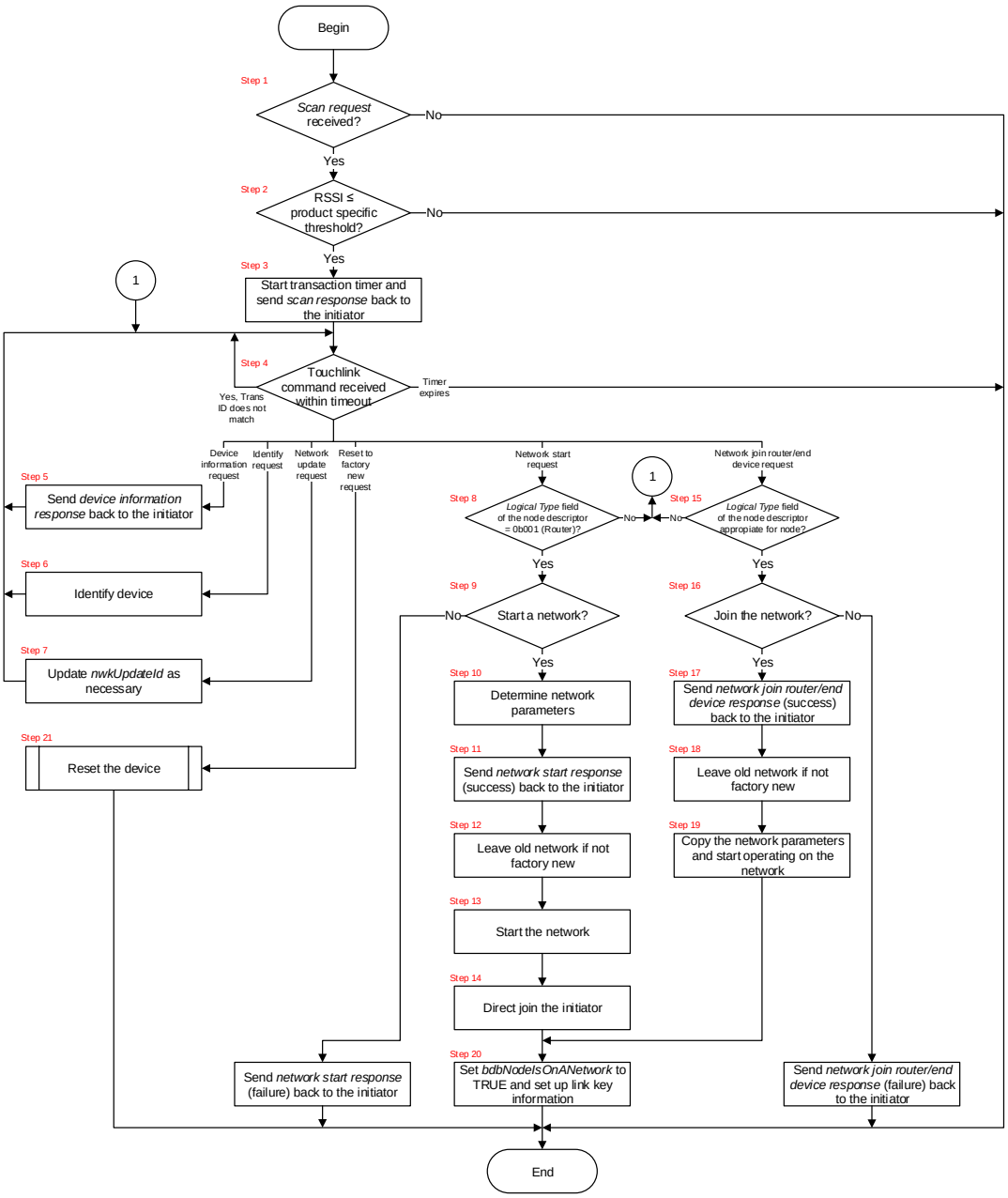
Note that simply accepting *touchlink commissioning* cluster *network start request* and *network join router/end device request* command frames could lead to undesired application behavior as the target leaves it current network and joins another network; this is known in touchlink as *stealing*. For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

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

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

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

1493



1494
1495

1496 **Figure 9 – Touchlink procedure for a target**

1497

1498 1. On receipt of a command other than the *touchlink commissioning* cluster *scan*
1499 *request* inter-PAN command frame, the target SHALL terminate the touchlink
1500 procedure for a target.
1501 2. The target sets *vIPTransID* to the value of the *inter-PAN transaction identifier*
1502 field and it SHALL determine whether to respond. If the *scan request*
1503 command was received with an RSSI less than or equal to a certain product
1504 specific threshold or the *link initiator* sub-field of the *touchlink information*

field is equal to 0, the target SHALL discard the frame and terminate the touchlink procedure for a target.

3. The target starts a timer for the current transaction to expire after *bdbcTLInterPANTransIdLifetime* seconds. The target SHALL then generate and unicast back to the initiator a *touchlink commissioning* cluster *scan response* inter-PAN command frame as follows. The *inter-PAN transaction identifier* field SHALL be set to *vIPTransID*. The *RSSI correction* field SHALL be set to a product specific RSSI correction value in order to compensate for RF signals losses between the radio and the outer side of a product; the initiator can then use this value in combination with the RSSI from each discovered target to select an appropriate target to continue with touchlink commissioning. The *touchlink priority request* sub-field of the *touchlink information* field SHALL be set to 1 if the target wishes to be considered as a priority by the initiator during touchlinking (e.g. if the target is power constrained and is responding to the scan following a button press from the user). The *response identifier* field SHALL be set to a random (non-sequential) value. If the *logical type* field of the node descriptor for the target is equal to 0b001 (ZigBee router) and *bdbNodeIsOnANetwork* is equal to TRUE, the *extended PAN identifier*, *network update identifier*, *logical channel*, *PAN identifier* and *network address* fields SHALL be set to the corresponding values of the network on which the target is currently operating. If the *logical type* field of the node descriptor for the target is not equal to 0b001 (ZigBee router) or *bdbNodeIsOnANetwork* is equal to FALSE, the *extended PAN identifier*, *network update identifier*, *logical channel*, *PAN identifier* and *network address* fields SHALL be set to zero. All other fields SHALL be set according to the specifics of the target.
4. On receipt of a *touchlink commissioning* cluster *device information request*, *identify request*, *network start request*, *network join router request*, *network join end device request* or *reset to factory new request* inter-PAN command frame with an *inter-PAN transaction identifier* field not equal to *vIPTransID*, the target SHALL discard the frame and continue from step 4. If the transaction timer expires, the target SHALL terminate the touchlink procedure for a target.
5. On receipt of a command other than the *device information request* inter-PAN command frame, the target SHALL continue from step 6. The target SHALL generate and unicast back to the initiator a *touchlink commissioning* cluster *device information response* inter-PAN command frame as follows. The *inter-PAN transaction identifier* field SHALL be set to *vIPTransID*. All other fields SHALL be set according to the specifics of the target. The target SHALL then continue from step 4.
6. On receipt of a command other than the *identify request* inter-PAN command frame, the target SHALL continue from step 8. The target SHALL identify

itself in an application specific way (e.g., by flashing a lamp) according to the value of the *identify time* field. No response SHALL be generated to an *identify request* inter-PAN command frame. The identify operation SHALL NOT block the target from receiving further commands. The target SHALL then continue from step 4.

7. On receipt of a command other than the *network update request* inter-PAN command frame, the target SHALL continue from step 7. If the *extended PAN identifier* and *PAN identifier* fields of the *network update request* inter-PAN command frame are not identical to its stored values or the *network update identifier* field is lower than or equal to *nwkUpdateId*, the target SHALL discard the frame and continue from step 4. If the *extended PAN identifier* and *PAN identifier* fields of the *network update request* inter-PAN command frame are identical to its stored values and the *network update identifier* field is higher than *nwkUpdateId*, the target SHALL update *nwkUpdateId* and its current logical channel with the values of the *network update identifier* and *logical channel* fields, respectively. The target SHALL then continue from step 4.
8. On receipt of a command other than the *network start request* inter-PAN command frame, the target SHALL continue from step 15. If the *logical type* field of the node descriptor is not equal to 0b001 (ZigBee router), the target SHALL discard the frame and continue from step 4.
9. The target SHALL decide by application specific means whether to allow itself to start a new network. If the target decides not to start a new network, it SHALL generate and unicast back to the initiator a *touchlink commissioning* cluster *network start response* inter-PAN command frame with the *inter-PAN transaction identifier* field set to *vIPTransID* and the *Status* field set to 0x01 (failure). The target SHALL then terminate the touchlink procedure for a target.
10. The target SHALL perform a network discovery to establish the network parameters. To do this, the target issues the *NLME-NETWORK-DISCOVERY.request* primitive to the NWK layer, with the *ScanChannels* parameter set either to correspond to the single *logical channel* field of the received *network start request* inter-PAN command frame if it is not equal to zero or to *bdbcTLPrimaryChannelSet* if it is equal to zero and the *ScanDuration* parameter set to *bdbScanDuration*. On receipt of the *NLME-NETWORK-DISCOVERY.confirm* primitive from the NWK layer, the target is notified of the results. Based on these results, the target SHALL select suitable values for the logical channel, PAN identifier and extended PAN identifier for the network.
11. The target SHALL generate and unicast back to the initiator a *network start response* inter-PAN command frame as follows. The *inter-PAN transaction identifier* field SHALL be set to *vIPTransID*. The *Status* field SHALL be set

to 0x00 (success). All other fields SHALL be set as appropriate to the verified network parameters.

12. If *bdbNodeIsOnANetwork* is equal to TRUE, the target SHALL perform a leave request on its old network. To do this, the target issues the *NLME-LEAVE.request* primitive to the NWK layer with the *DeviceAddress* parameter set to NULL, the *RemoveChildren* parameter set to FALSE and the *Rejoin* parameter set to FALSE. On receipt of the *NLME-LEAVE.confirm* primitive, the target is notified of the status of the request to leave the network. The target SHALL then clear all ZigBee persistent data (see sub-clause 6.9) except the outgoing NWK frame counter.
13. The target SHALL then copy the new network parameters to its network information base and start operating on the new network. To do this, the target issues the *NLME-START-ROUTER.request* primitive to the NWK layer with the *BeaconOrder* parameter set to 0x0f, the *SuperframeOrder* set to 0x00 and the *BatteryLifeExtension* parameter set to FALSE. On receipt of the *NLME-START-ROUTER.confirm* primitive, the target is notified of the status of the request to start.
14. The target SHALL perform a direct join on behalf of the initiator. To do this, the target issues the *NLME-DIRECT-JOIN.request* primitive to the NWK layer with the *DeviceAddress* parameter set to the IEEE address of the initiator. On receipt of the *NLME-DIRECT-JOIN.confirm* primitive, the target is notified of the status of the direct join request. The target SHALL then continue from step 20.
15. On receipt of a command other than the *network join router request* or a *network join end device* inter-PAN command frame, the target SHALL continue from step 21. If a *network join router request* inter-PAN command frame was received and the *logical type* field of the node descriptor is not equal to 0b001 (ZigBee router) or a *network join end device* inter-PAN command frame was received and the *logical type* field of the node descriptor is not equal to 0b010 (ZigBee end device), the target SHALL discard the frame and continue from step 4.
16. The target SHALL decide by application specific means whether to allow itself to be joined to another network. If the target decides not to be joined to another network, it SHALL generate and unicast back to the initiator a corresponding *touchlink commissioning* cluster *network join router response* or *network join end device response* inter-PAN command frame, depending on whether a *network join router request* or *network join end device request* inter-PAN command frame, respectively, was received with the *inter-PAN transaction identifier* field set to *vIPTransID* and the *Status* field set to 0x01 (failure). The target SHALL then terminate the touchlink procedure for a target.

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

9 Reset

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

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

9.1 Reset via the basic cluster

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

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

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

9.2 Reset via the touchlink commissioning cluster

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

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

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

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

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

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

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

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

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

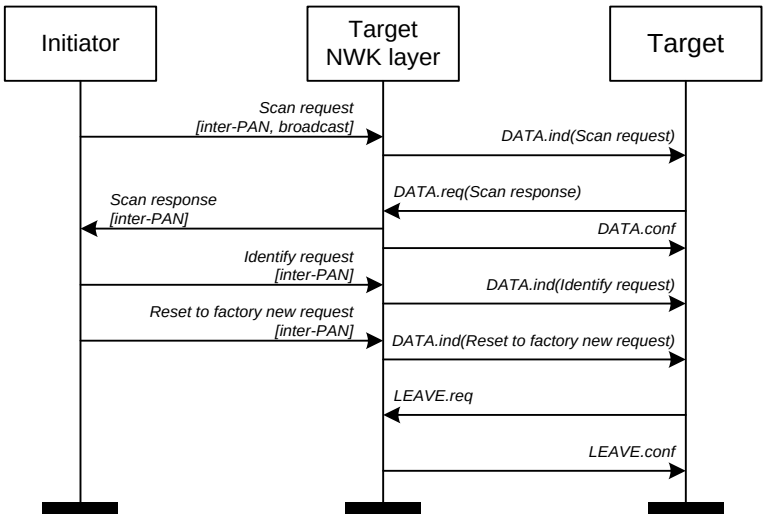


Figure 10 – Resetting a target to factory new via the *touchlink commissioning* cluster

9.3 Reset via the network leave command

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

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

9.4 Reset via Mgmt_Leave_req ZDO command

ZigBee-PRO provides an *Mgmt_Leave_req* ZDO command which is designed to request that a remote node leaves the network by clearing all ZigBee persistent data

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

The *Mgmt_Leave_req* ZDO command is specified in sub-clause 2.4.3.3.5 of [R1].

9.5 Reset via a local action

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

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

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

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

10 Security

10.1 Install codes

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

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

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.

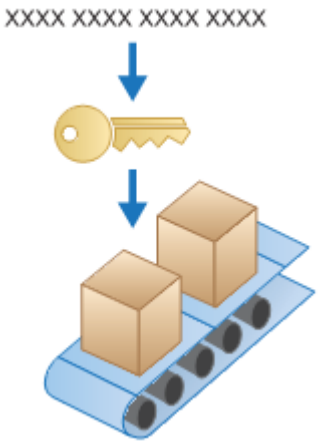


Figure 11 – Node Install Code process

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

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.

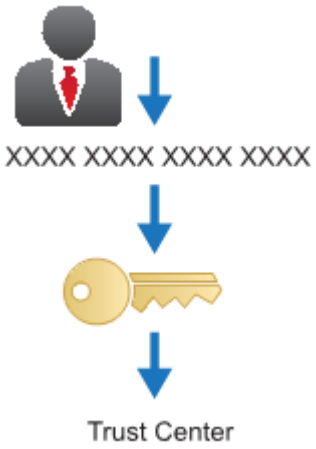


Figure 12 – Install code use with the Trust Center

10.1.1 Install code format

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

Example:

Install code of "83FE D340 7A93 9723 A5C6 39B2 6916 D505 C3B5"

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

10.1.1.1 CRC algorithm information

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

Length: 16
Polynomial: $x^{16} + x^{12} + x^5 + 1$ (0x1021)
Initialization method: Direct
Initialization value: 0xFFFF
Final XOR value: 0xFFFF
Reflected In: True
Reflected Out: True

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

10.1.2 Hashing Function

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

Install code example:

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

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

10.1.2.1 MMO hash code example

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

10.2 Node operations

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

10.2.1 Joining node policy values

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

10.2.1.1 *acceptNewUnsolicitedTrustCenterLinkKey* policy

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

Note this value is ignored in a distributed security network.

10.2.1.2 *acceptNewUnsolicitedApplicationLinkKey* policy

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

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

10.2.2 Trust Center address

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

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

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

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

10.2.3 Trust Center Link Keys

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

10.2.4 Requesting a Link Key

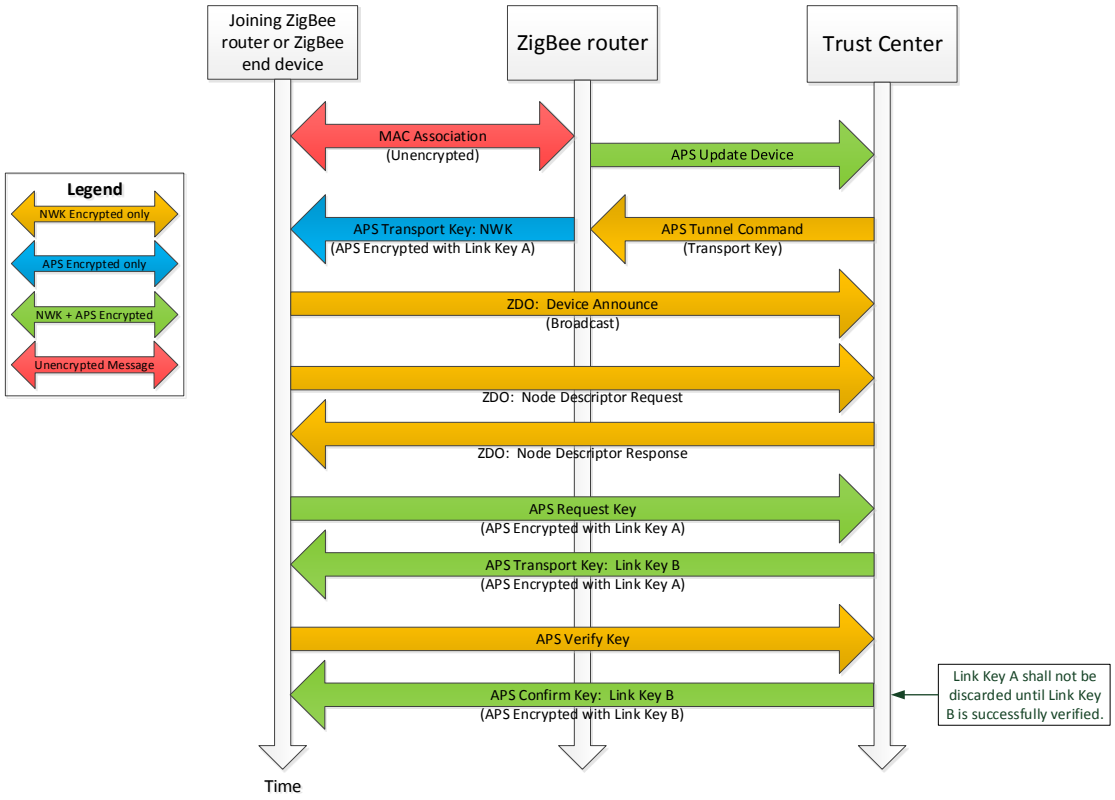
If *bdbTCLinkKeyExchangeMethod* is equal to 0x00, the node SHALL exchange its initial link key with one generated by the Trust Center as part of its initial joining operations in a centralized security network.

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

10.2.5 Trust Center link key exchange procedure

This section defines the procedure to retrieve a new Trust Center link key for a node. A sequence chart for this procedure showing the messages exchanged and the 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

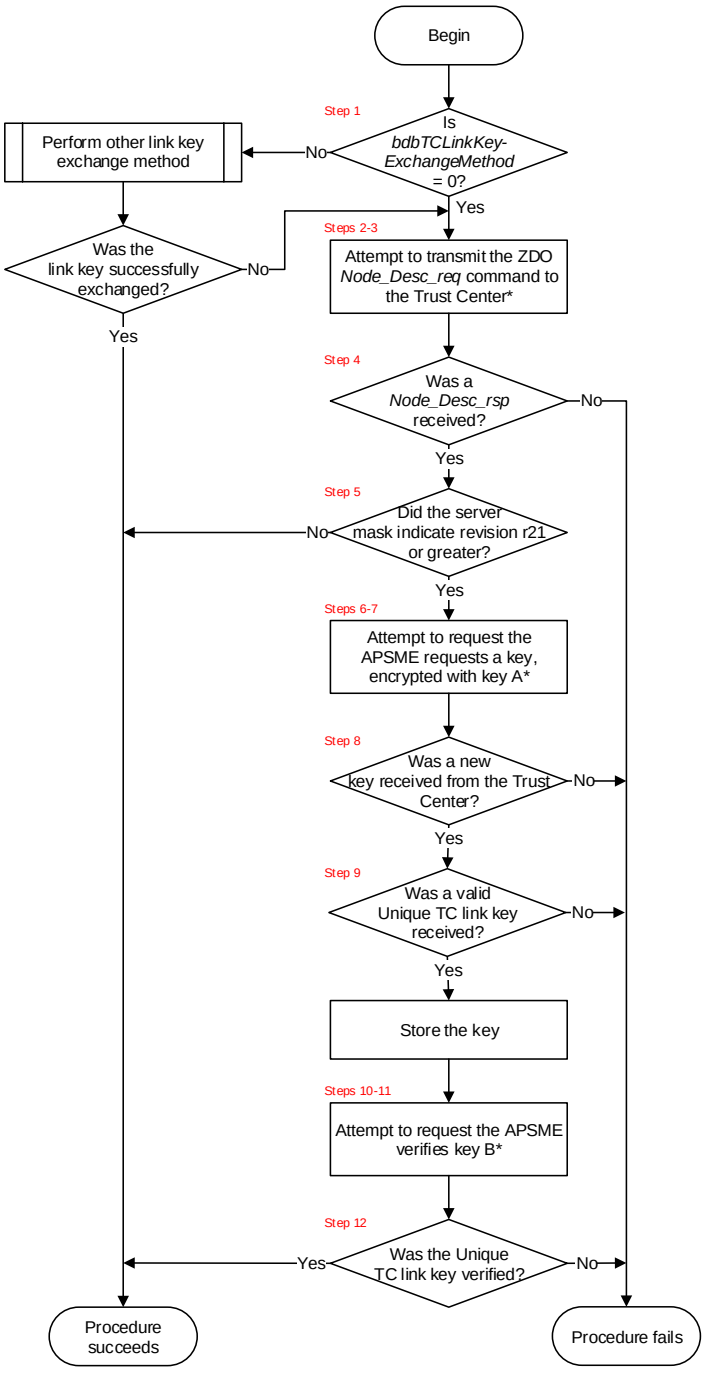


Figure 14 – Trust Center link key exchange procedure

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.

2. The joining node sets *bdbTCLinkKeyExchangeAttempts* to 0.
3. The joining node SHALL send a ZDO *Node_Desc_req* command to the Trust Center. It then starts a timer of *bdbcTCLinkKeyExchangeTimeout* seconds and increments *bdbTCLinkKeyExchangeAttempts* by 1.
4. If a ZDO *Node_Desc_rsp* command is not received before the timer expires, the joining node SHALL determine whether to retry the attempt as follows:
    a. If *bdbTCLinkKeyExchangeAttempts* is less than *bdbTCLinkKey-ExchangeAttemptsMax*, the joining node SHALL continue from step 3.
    b. If *bdbTCLinkKeyExchangeAttempts* is equal to *bdbTCLinkKey-ExchangeAttemptsMax* the joining node SHALL terminate the Trust Center link key exchange procedure with a failure status.
5. If the *server mask* field of the receiver *node* descriptor indicates a stack revision of r20 or earlier, the joining node SHALL terminate the Trust Center link key exchange procedure with a success status.
6. The joining node sets *bdbTCLinkKeyExchangeAttempts* to 0.
7. The joining node SHALL request a new link key from the Trust Center. To do this, the joining node issues an *APSME-REQUEST-KEY.request* primitive encrypted with its initial Trust Center link key (key A). It then starts a timer of *bdbcTCLinkKeyExchangeTimeout* seconds and increment *bdbTCLinkKeyExchangeAttempts* by 1.
8. If the joining node does not receive an *APSME-TRANSPORT-KEY.indication* primitive before the timer expires, the joining node SHALL determine whether to retry the attempt as follows:
    a. If *bdbTCLinkKeyExchangeAttempts* is less than *bdbTCLinkKey-ExchangeAttemptsMax*, the joining node SHALL continue from step 7.
    b. If *bdbTCLinkKeyExchangeAttempts* is equal to *bdbTCLinkKey-ExchangeAttemptsMax*, the joining node SHALL terminate the Trust Center link key exchange procedure with a failure status.
9. The joining node SHALL find the entry in the *apsDeviceKeyPairSet* with a *DeviceAddress* that corresponds to the *apsTrustCenterAddress*. If the *KeyType* parameter of the received *APSME-TRANSPORT-KEY.indication* primitive is not equal to 0x04 (Unique Trust Center Link Key) or the link key contained in the primitive is identical to the *LinkKey* value of the *apsDeviceKeyPairSet* entry, the joining node SHALL terminate the Trust Center link key exchange procedure with a failure status. Otherwise, the joining node SHALL replace the *LinkKey* value with the key contained in the primitive (link key B), it MAY then set *OutgoingFrameCounter* to 0 and it SHALL set the *IncomingFrameCounter* to 0 for the *apsDeviceKeyPairSet* entry.
10. The joining node sets *bdbTCLinkKeyExchangeAttempts* to 0.
11. The joining node SHALL verify the new link key with the Trust Center. To do this, the joining node issues an *APSME-VERIFY-KEY.request* primitive to

verify the new key (link key B). It then starts a timer of *bdbcTCLink-KeyExchangeTimeout* seconds and increment *bdbTCLinkKeyExchange-Attempts* by 1.

12. If the joining node does not receive an *APSME-CONFIRM-KEY.indication* primitive before the timer expires, the joining node SHALL determine whether to retry the attempt as follows:
    a. If *bdbTCLinkKeyExchangeAttempts* is less than *bdbTCLinkKey-ExchangeAttemptsMax*, the joining node SHALL continue from step 11.
    b. If *bdbTCLinkKeyExchangeAttempts* is equal to *bdbTCLinkKey-ExchangeAttemptsMax*, the joining node SHALL terminate the Trust Center link key exchange procedure with a failure status.
13. The joining node SHALL terminate the Trust Center link key exchange procedure with a success status.

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

10.2.6 Receiving new Link Keys

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

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

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

10.3 Trust Center behavior

10.3.1 Adding the install code

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

DeviceAddress set to the EUI64 of the joining node and the *LinkKey* value equal to the installation code link key.

i. The *apsLinkKeyType* of that entry SHALL be set to 0x00 (Unique). See Table 4.39 in [R1].

b. If the Trust Center does not require use of installation code link key then it shall create a corresponding entry in its AIB *apsDeviceKeyPairSet* when the node joins the network.

10.3.2 Adding a new node into the network

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

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

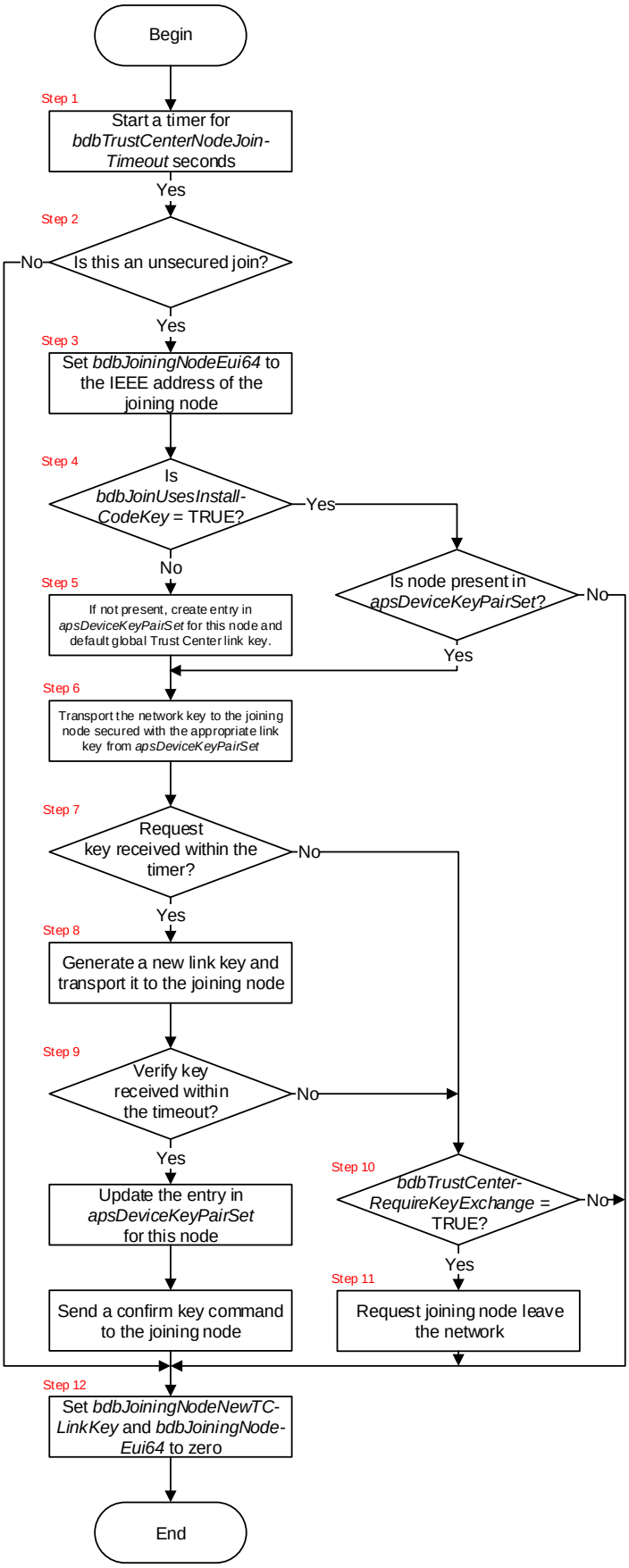


Figure 15 – Trust Center link key exchange procedure

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 *bdbJoiningNode-Eui64* it SHALL do the following.
    a. It SHALL find the entry in the *apsDeviceKeyPairSet* where the *DeviceAddress* corresponds to the *bdbJoiningNodeEui64*.

b. If the value of *bdbJoiningNodeNewTCLinkKey* is different than the value of the *LinkKey* of the *apsDeviceKeyPairSet* entry, the Trust Center:
    i. MAY set *OutgoingFrameCounter* to 0 and SHALL set *IncomingFrameCounter* to 0 within the *apsDeviceKeyPairSet* entry.
    ii. SHALL copy the *bdbJoiningNodeNewTCLinkKey* value to the LinkKey value of the *apsDeviceKeyPairSet.*
c. It SHALL issue the APSME-CONFIRM-KEY.request primitive with the *DestAddress* field set to *bdbJoiningNodeEui64*.
d. It SHALL then continue from step 12.

10. If *bdbTrustCenterRequireKeyExchange* is equal to FALSE (the link key does not have to be exchanged), the Trust Center SHALL continue from step 12.
11. The Trust Center SHALL request that the joining node leave the network. To do this, the Trust Center issues the *APSME-REMOVE-DEVICE.request* primitive with the *ParentAddress* parameter set to the *SrcAddress* parameter from the *APSME-UPDATE-DEVICE.indication* primitive, received in step 1, and the *ChildAddress* parameter set to *bdbJoiningNodeEui64*.
12. The Trust Center SHALL do the following before terminating the procedure for adding a new node into the network:
    a. Expire the *bdbTrustCenterNodeJoinTimeout* timer.
    b. Set the value of the *bdbJoiningNodeNewTCLinkKey* to zero.
    c. Set the value of the *bdbJoiningNodeEui64* to zero.

10.3.3 Behavior when a known node joins

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

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

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

10.4 Distributed security network behavior

10.4.1 Adding a new node into the network

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

11 Annex A: Recommended practices

11.1 Recommendations for centralized commissioning

11.1.1 Centralized commissioning overview

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

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

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

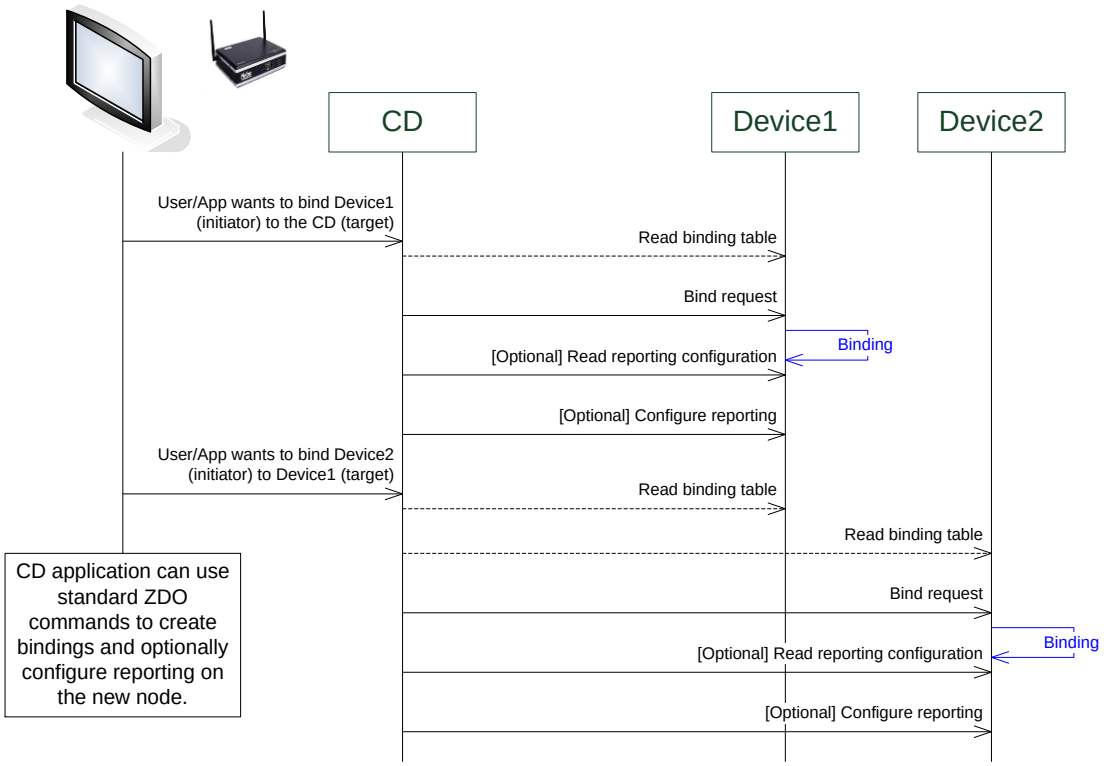


Figure 16 – Principle of centralized commissioning with a commissioning director

11.1.2 Recommendations for device discovery

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

11.1.2.1 New nodes joining

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

11.1.2.2 Nodes in existing network

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

There are several ways for a CD to discover nodes in the network but it is RECOMMENDED that the CD uses the *Mgmt_Lqi_req* ZDO command. The benefits of using *Mgmt_Lqi_req* (instead of *IEEE_addr_req* or *NWK_addr_req*) are listed below:

- ZigBee logical device type information of ZigBee coordinator, ZigBee router, ZigBee end device
- Rx_On_when_Idle information
- Information about parent-child relationships

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

11.1.2.3 Establishing communications with end devices

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