Zigbee Specification

Revision 23

Zigbee Document 05-3474-23	
March 15, 2023	
Sponsored by: Connectivity Standards Alliance	
Accepted by	This document has been accepted for release by the Alliance Board of Directors.
Abstract	The Zigbee Specification describes the infrastructure and services available to applications operating on the Zigbee platform.
Keywords	Zigbee, Stack, Network, Application, Profile, Framework, Device Description, Binding, Security

This page intentionally left blank.

Notice of Use and Disclosure

Connectivity Standards Alliance

508 Second Street
Suite 206
Davis, CA 95616
USA

This page intentionally left blank.

Document History

Zigbee Specification History

Revision	Date	Description
	December 14, 2004	Zigbee v.1.0 draft ratified
r06	February 17, 2006	Zigbee Specification (Zigbee document number 053474r06/07) incorporating errata and clarifications: Zigbee document numbers 053920r02, 053954r02, 06084r00, and 053474r07
r07	April 28, 2006	Changes made per Editorial comments on spreadsheet
r13	October 9, 2006	Zigbee-2006 Specification (see letter ballot comments and resolution in Zigbee document 064112)
r14	November 3, 2006	Zigbee-2007 Specification (adds features described in 064270, 064269, 064268, 064281, 064319, and 064293)
r15	December 12, 2006	Zigbee-2007 Specification incorporating errata and clarifications: 074746
r16	May 31, 2007	Zigbee-2007 Specification incorporating errata and clarifications: 07819
r17	October 19, 2007	Zigbee-2007 specification incorporating errata: 075318, 075053, 075164, 075098
r18	June 16, 2009	Zigbee-2007 specification incorporating errata: 08012
r19	September 28, 2010	Zigbee-2007 specification incorporating errata described in document 105413r04
r20	September 18, 2012	Zigbee-2007 specification incorporating errata described in 11-53778-r13 and 12-0030-01
r21	August 5, 2015	Zigbee specification incorporating large updates as follows: 1. Chapter 2 – Application Layer a. Addition of Parent Announce ZDO message b. Addition of over-the-air mechanism for detecting device’s implemented specification version. 2. Chapter 3 – Network Layer a. Add End device timeout protocol and aging mechanism 3. Chapter 4 – Security a. Removal of High Security b. Addition of Trust Center Link Key update services c. Cleanup of frame counter handling, d. Addition of Distributed Trust Center mode 4. Annex D – MAC And PHY Sub-layer Clarifications a. Update to IEEE Std 802.15.4-2011 5. Annex G – Inter-PAN a. Formalization of Inter-PAN frame formats and service handling.

Revision	Date	Description
		6. Annex H – Inter-PAN Test Vectors of Green Power Inter-PAN test vectors.
r22 0.7	July 25 2016	Additional functionality to support sub GHz FSK PHY/MAC 0.7 Reballot comments included
R22 0.9	Sep 30 2016	R21 errata and other critical CCBs added. PHY/MAC spec integrated.
R22 0.9	December 1st 2016	Updated with comments and issued for recirculation ballot.
R22 1.0	March 20, 2017	Updated with reballot comments and issues for draft rev 1.0 release.
R23 0.5	July 4, 2018	Chapter 3: Clean-up
R23 0.5	September 28, 2018	Added Dynamic Link Key NFR text (up through chapter 3).
R23 0.5	October 8, 2018	Included Curve25519 text, Low-power (CSL) changes. Included most of the WWAH items.
R23 0.5	November 25, 2018	Integrated ZDO deprecation and remove all references to caches.
R23 0.5	December 03, 2018	Removed the User descriptor and complex descriptor. Update the use of Allocated Address bit.
R23 0.5	January 11, 2019	Integrated Sub Gig routing and regional Sub Gig annex.
R23 0.5	January 15, 2019	Integrated Routing updates.
R23 0.5	January 28, 2019	Joining/Rejoining/Dynamic Link key negotiation updates
R23 0.7	May 29, 2019	Comment resolution from R23 0.5 ballot
R23 0.6	January 22, 2020 Rev 9	Merged changes from a separate 0.8 document into this. Updated Clear All Bindings and Security Decommission Req. Updated Security Key Negotiation Req and Security Key Negotiation Rsp. Added section 1.2.6.
R23 0.6	January 24, 2020 Rev 10	Updated Annex I based on changes from Sheffield Face-to-face. Merged from conflicted document on Causeway. Reinstated Symmetric Passphrase, Next Channel Change, and Next PAN ID Global TLV. Updated rules on malformed TLVs and various other minor edits proposed in Sheffield.
R23 0.6	January 30, 2020 Revision 11	Comment fixes: 2554, 2555, 2621, 2561, 2623, 3111
R23 0.6	February 4, 2020 Revision 12	Updated SEC_Get_Authentication_token_req to use TLVs. Updated SEC_Get_Authentication_Level_req to use TLVs. Updated Security_Get_Authentication_Level_rsp to use TLVs.
R23 0.6	February 20, 2020 Revision 13	Updated Annex J (aligned KDF between J.1 and J.2, added H*(x), Curve25519 Private Key Clamping, endianness clarifications).

Revision	Date	Description
R23 0.6	February 27, 2020 Revision 14	Addressed comments for sections 1.1.5, 1.2.5 and added in APS Frame Counter Synchronization.
R23 0.6	March 8, 2020 Revision 15	Addressed comments for sections 2.3.2.4 and 2.4. Removed Beacon Appendix version from Beacon Payload. Added Router Information TLV.
R23 0.6	March 16, 2020 Revision 16	Made Annex I changes. Added Fragmentation Parameters Global TLV, Joiner Encapsulation Global TLV, Beacon Appendix Encapsulation Global TLV. Section 1.2.6, 2.4.3.4
R23 0.6	March 22, 2020 Revision 17	Added the Beacon Survey Configuration TLV to Mgmt_NWK_Beacon_Survey_req. Updated sections 2.4.2.8.3, 2.4.3.4, 4.7.3.12.
R23 0.6	April 6, 2020 Revision 18	Added Annex C.7 with test vectors for the key agreement schemes defined in Annex J: • ECDHE-PSK/P-256/SHA-256/HMAC-SHA-256-128 (C.7.1) • SPEKE/Curve25519/AES-MMO-128/HMAC-AES-MMO-128 (C.7.2)
R23 0.6	April 6, 2020 Revision 19	Updated Security_Decommission_Req, Security_Start_Key_Negotiation_Req and Secuity_Get_Authentication_Token_Req. Removed 2.5.4.6 (Device and Service Discovery), 2.5.4.7 (Security Manager) and 2.5.4.8 (Binding Manager).
R23 0.6	April 15, 2020 Revision 20	Updated Key negotiation before joining diagram (Section 1.1.5.4). Updated sections 3.2.2, 3.4.
R23 0.6	April 30, 2020 Revision 21	Updated sections 3.4, 3.6.1.4, 3.6.1.5,3.6.4. CCB 3190
R23 0.6	May 15, 2020 Revision 22	Updated section 3.6.1.4. Updated sections 4.4.2, 4.4.9, 4.4.12, 4.6.
R23 0.6	May 21, 2020 Revision 23	Updated section 2.4.3.3.12, 3.6.9.2. Updated the Beacon Survey Configuration TLV. Added in Multi-hop Dynamic Link Key Changes. Updated section 2.4.3.4.1.3 Effect on receipt (of ZDO Security_Key_Negotiation_req). Updated section 2.4.4.5.1.6 Effect on Receipt [ZDO Security_Start_Key_Negotiation_rsp]. Updated section 3.4.14.3.2 TLV [NWK Commissioning Request Command]. Added section 4.4.10 Key Negotiation Services. Added section 4.4.12.9 Relay Message Downstream. Added section 4.4.12.10 Relay Message Upstream. Added sections 4.6.3.8.4 – 4.6.3.8.12.

Revision	Date	Description
R23 0.6	June 19,2020 Revision 24	Removed section 2.4.4.1.1. Added in CCB 2673. Updated sections 2.4.3.3.7, 4.9.
R23 0.6	June 26,2020 Revision 25	Updated session identifier for SPEKE simplified (Annex J) and test vector (Annex C). Added PSK capabilities and selection to key negotiation. PSK enumerations for Z3BLE allocated Clarified usage of APS acknowledgments in APS commands. Added section 4.9.7. Addressed comments 2802, 2988,2989,3042,3080. Fixed endian issue for the Test Vectors (Curve25519 and P-256).
R23 0.7	August 28, 2020 Revision 26	Updated based on the 0.5 ballot editorial comments.
R23 0.9	June 8, 2022	Various integrations of 0.9 ballot comments: ZPC-1181 ZPC-1055 ZPC-1036 ZPC-1031 ZPC-1027 ZPC-1022 ZPC-1014 ZPC-1013 ZPC-1012 ZPC-1009 ZPC-1004 ZPC-1003 ZPC-1002 ZPC-1001 ZPC-1000 ZPC-999 ZPC-998 ZPC-996 ZPC-995 ZPC-994 ZPC-993 ZPC-992 ZPC-991 ZPC-990 ZPC-975 ZPC-974 ZPC-972 ZPC-969 ZPC-968 ZPC-967 ZPC-966 ZPC-965 ZPC-964 ZPC-963 ZPC-962 ZPC-961 ZPC-960 ZPC-956 ZPC-955 ZPC-954 ZPC-953 ZPC-952 ZPC-948 ZPC-947 ZPC-946 ZPC-945 ZPC-944 ZPC-943 ZPC-942 ZPC-941 ZPC-940 ZPC-939 ZPC-938 ZPC-937 ZPC-936 ZPC-935 ZPC-934 ZPC-933 ZPC-932 ZPC-931 ZPC-929 ZPC-928 ZPC-927 ZPC-923 ZPC-922 ZPC-920 ZPC-919 ZPC-918 ZPC-917 ZPC-913 ZPC-912 ZPC-911 ZPC-910 ZPC-909 ZPC-908 ZPC-905 ZPC-904 ZPC-903 ZPC-901 ZPC-900 ZPC-898 ZPC-897 ZPC-896 ZPC-895 ZPC-894 ZPC-893 ZPC-892 ZPC-891 ZPC-890 ZPC-889 ZPC-888 ZPC-885

Revision	Date	Description			
		ZPC-884	ZPC-882	ZPC-881	ZPC-880
		ZPC-879	ZPC-878	ZPC-877	ZPC-876
		ZPC-875	ZPC-873	ZPC-869	ZPC-868
		ZPC-867	ZPC-864	ZPC-863	ZPC-861
		ZPC-860	ZPC-855	ZPC-851	ZPC-844
		ZPC-842	ZPC-840	ZPC-838	ZPC-836
		ZPC-833	ZPC-831	ZPC-827	ZPC-826
		ZPC-824	ZPC-818	ZPC-812	ZPC-811
		ZPC-809	ZPC-803	ZPC-795	ZPC-792
		ZPC-781	ZPC-775	ZPC-752	ZPC-762
		ZPC-588	ZPC-556	ZPC-546	ZPC-486
		ZPC-468	ZPC-330	ZPC-315	ZPC-295
		ZPC-264	ZPC-249	ZPC-180	ZPC-12
		ZPC-10	ZPC-4		
R23 0.95	August 15, 2022	Various integrations of 0.95 ballot comments:			
		ZPC-1302	ZPC-1301	ZPC-1300	ZPC-1299
		ZPC-1298	ZPC-1297	ZPC-1267	ZPC-1214
		ZPC-1259	ZPC-1247	ZPC-1242	ZPC-1234
		ZPC-1231	ZPC-1224	ZPC-1220	ZPC-1218
		ZPC-1217	ZPC-1181	ZPC-1180	ZPC-1178
		ZPC-1170	ZPC-1156	ZPC-1155	ZPC-1154
		ZPC-1153	ZPC-1152	ZPC-1151	ZPC-1150
		ZPC-1149	ZPC-1148	ZPC-1147	ZPC-1146
		ZPC-1145	ZPC-1144	ZPC-1143	ZPC-1142
		ZPC-1141	ZPC-1140	ZPC-1139	ZPC-1138
		ZPC-1137	ZPC-1136	ZPC-1135	ZPC-1134
		ZPC-1133	ZPC-1132	ZPC-1131	ZPC-1130
		ZPC-1129	ZPC-1128	ZPC-1127	ZPC-1126
		ZPC-1125	ZPC-1124	ZPC-1123	ZPC-1122
		ZPC-1121	ZPC-1120	ZPC-1119	ZPC-1118
		ZPC-1117	ZPC-1116	ZPC-1115	ZPC-1114
		ZPC-1113	ZPC-1112	ZPC-1111	ZPC-1110
		ZPC-1109	ZPC-1108	ZPC-1107	ZPC-1106
		ZPC-1105	ZPC-1104	ZPC-1102	ZPC-1101
		ZPC-1100	ZPC-1099	ZPC-1098	ZPC-1097

Revision	Date	Description
		ZPC-1096 ZPC-1095 ZPC-1094 ZPC-1093 ZPC-1092 ZPC-1091 ZPC-1090 ZPC-1089 ZPC-1088 ZPC-1087 ZPC-1086 ZPC-1085 ZPC-1084 ZPC-1083 ZPC-1082 ZPC-1081 ZPC-1080 ZPC-1079 ZPC-1078 ZPC-1077 ZPC-1076 ZPC-1075 ZPC-1074 ZPC-1073 ZPC-1072 ZPC-1071 ZPC-1070 ZPC-1069 ZPC-1068 ZPC-1067 ZPC-1066 ZPC-1065 ZPC-1064 ZPC-1063 ZPC-1062 ZPC-1060 ZPC-1059 ZPC-1058 ZPC-1057 ZPC-1056 ZPC-1054 ZPC-1053 ZPC-1052 ZPC-1033 ZPC-1032 ZPC-1011 ZPC-835
r23	11 Jan 2023	NCR review completed and Security audit completed.

Table of Contents

List of Tables

List of Figures

This page intentionally left blank.

CHAPTER 1. ZIGBEE PROTOCOL OVERVIEW

1.1 Protocol Description

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

1.1.1 Scope

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

1.1.2 Purpose

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

1.1.3 Stack Architecture

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

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

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

The MAC sub-layer controls access to the radio channel using either a CSMA-CA or LBT mechanism, depending on the underlying MAC/PHY. Its responsibilities may also include transmitting beacon frames, synchronization, and providing a reliable transmission mechanism. A complete description of the IEEE Std 802.15.4 MAC sub-layer can be found in [B1]. Figure 1-1 represents the outline of the Zigbee Stack Architecture.

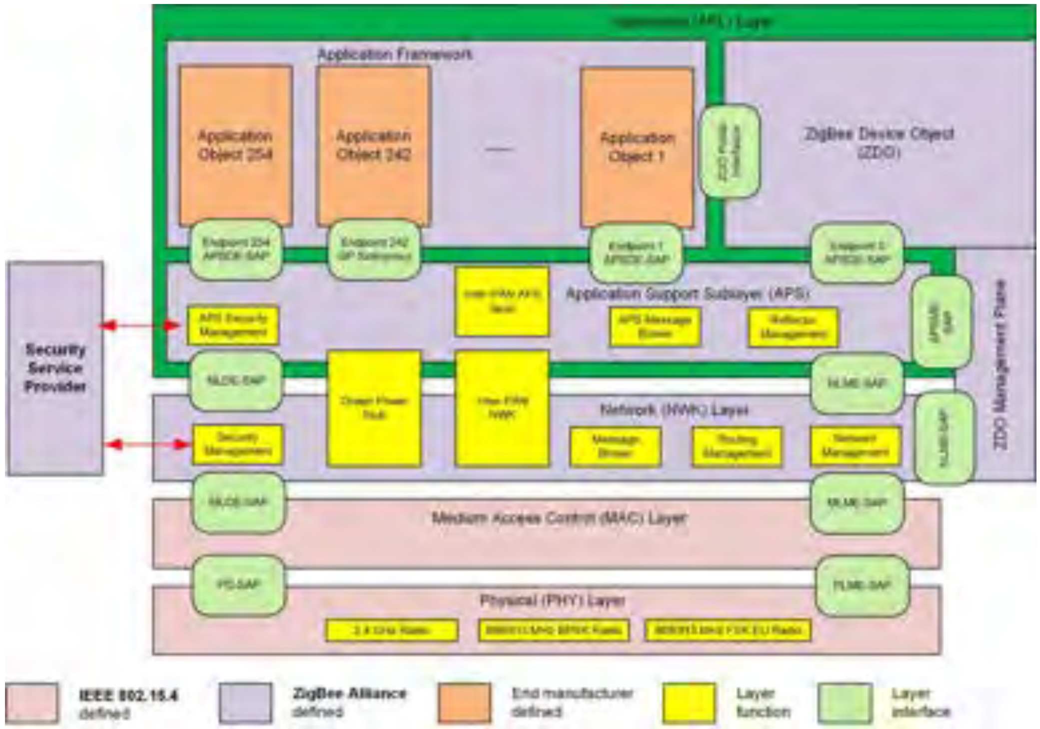


Figure 1-1. Outline of the Zigbee Stack Architecture

1.1.3.1 Non-Certifiable Features

Some Zigbee functions are not certifiable on every combination of MAC/PHY. These are listed below:

a) Green Power is certifiable only on the 2.4GHz O-QPSK PHY

b) Inter-PAN is certifiable only on the 2.4GHz O-QPSK PHY

1.1.4 Network Topology

The Zigbee network layer (NWK) supports star and mesh topologies. In a star topology, the network is controlled by one single device called the Zigbee coordinator. The Zigbee coordinator is responsible for initiating and maintaining the devices on the network. All other devices, known as end devices, directly communicate with the Zigbee coordinator. In mesh topologies, the Zigbee coordinator is responsible for starting the network and for choosing certain key network parameters, but the network may be extended through the use of Zigbee routers. Mesh networks allow full peer-to-peer communication. Zigbee routers in mesh networks do not currently emit regular IEEE Std 802.15.4 beacons. This specification describes only intra-PAN networks, that is, networks in which communications begin and terminate within the same network. An exception is the inter-PAN feature which allows the Zigbee stack to be bypassed, for example to initialize Zigbee network settings out of band.

1.1.5 Overall Joining Flow

This section will describe the high-level flow for a device joining the network. This flow varies due to changes over time in the specification and is highlighted below.

For a device to operate as a fully authorized device on the Zigbee mesh network it must have the current network key. In centralized networks the role of the Trust Center acts as gatekeeper to determine when and who is authorized to join the network. The joining flows below show a multi-hop join with three devices: joiner, router, and trust center. When only two devices are present, Trust Center and joiner, the roles of Trust center and Router are collapsed together, and communication is handled internal to the device.

Instead of the centralized security model, which requires a trust center and coordinator to form and manage the network, a decentralized approach is also available called a Distributed Security Network. This allows any router to authorize a new device by sending the network key directly to it without consulting a centralized authority.

In Revision 21 and earlier all Zigbee devices joining the network are pre-configured with a Trust Center Link Key. This may either be a well-known default link key or a secret, device-specific key known as an Install Code derived Key. Joining with the well-known Trust Center link key represents a moment of insecurity that may be acceptable to the Trust Center managing the network's security. Alternatively, the Trust Center may be configured to require use of the Install Code derived key when joining the network. In that case all devices must pass their Install Code derived key out-of-band to the Trust Center prior to joining.

Revision 23 introduces a new mechanism that utilizes public key cryptography to securely negotiate a key before receiving the network key.

Note that all diagrams in this section do not show the APS Acks but they are present on the ZDO messages. Also, fragmentation may occur but that is not shown in these diagrams.

1.1.5.1 Static Key Joining

Static Key Joining is when both the joining device and Trust Center have been configured with a fixed link key. This key can be one of several standardized values listed below.

1. Default Global Trust Center link key (ZigBeeAlliance09)
2. Install code derived pre-configured link key
3. Distributed Security Global Link Key
4. Touchlink Preconfigured Link Key

Zigbee devices in Revision 20 and earlier had only a joining flow that involved the following sequence. This is shown in Figure 1-2.

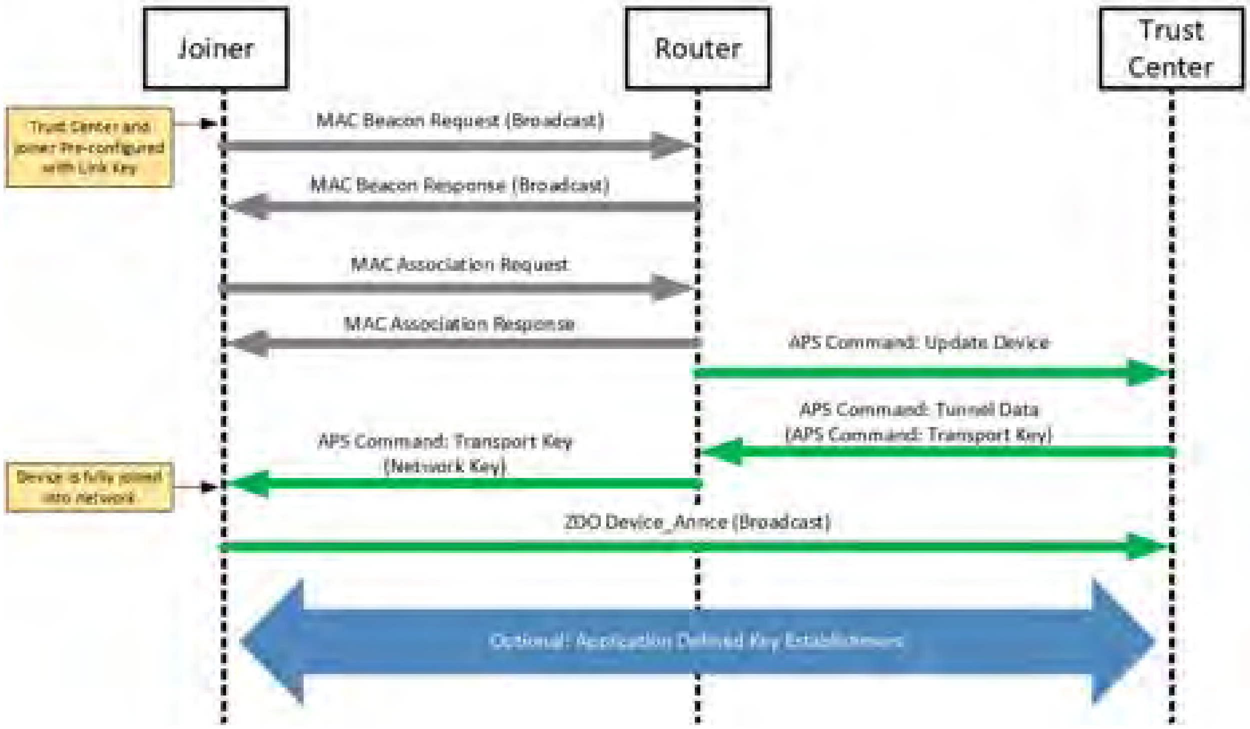


Figure 1-2. Static Key Joining without Updating Trust Center Link Key

Devices in Revision 20 and earlier only updated their pre-configured link key when an application defined key establishment protocol was used, such as the Key Establishment Cluster in the Zigbee Smart Energy Specification.

1.1.5.2 Joining and Using Key Assignment

With Revision 21 of the specification and beyond, the joiner is required to replace its initial trust center link key with an updated key. The joiner uses Static Key Joining to initially gain access to the network, but then performs a Trust Center Link Key Update. The mechanism for updating the trust center link key can utilize stack primitives or use a higher layer protocol. If no application defined protocol is performed, devices can use the Key Assignment Mechanism.

In Key Assignment a new symmetric link key is requested by the joiner and chosen by the Trust Center and the exchange is shown in Figure 1-3. This update occurs regardless of whether the initial trust center link key was a well-known or Install Code derived key. This prevents an attacker that obtains the trust center link key after the join from using it to gain access to the network. This joining flow was very similar to the previous flow but added additional steps after receiving the network key to request a trust center link key from the trust center.

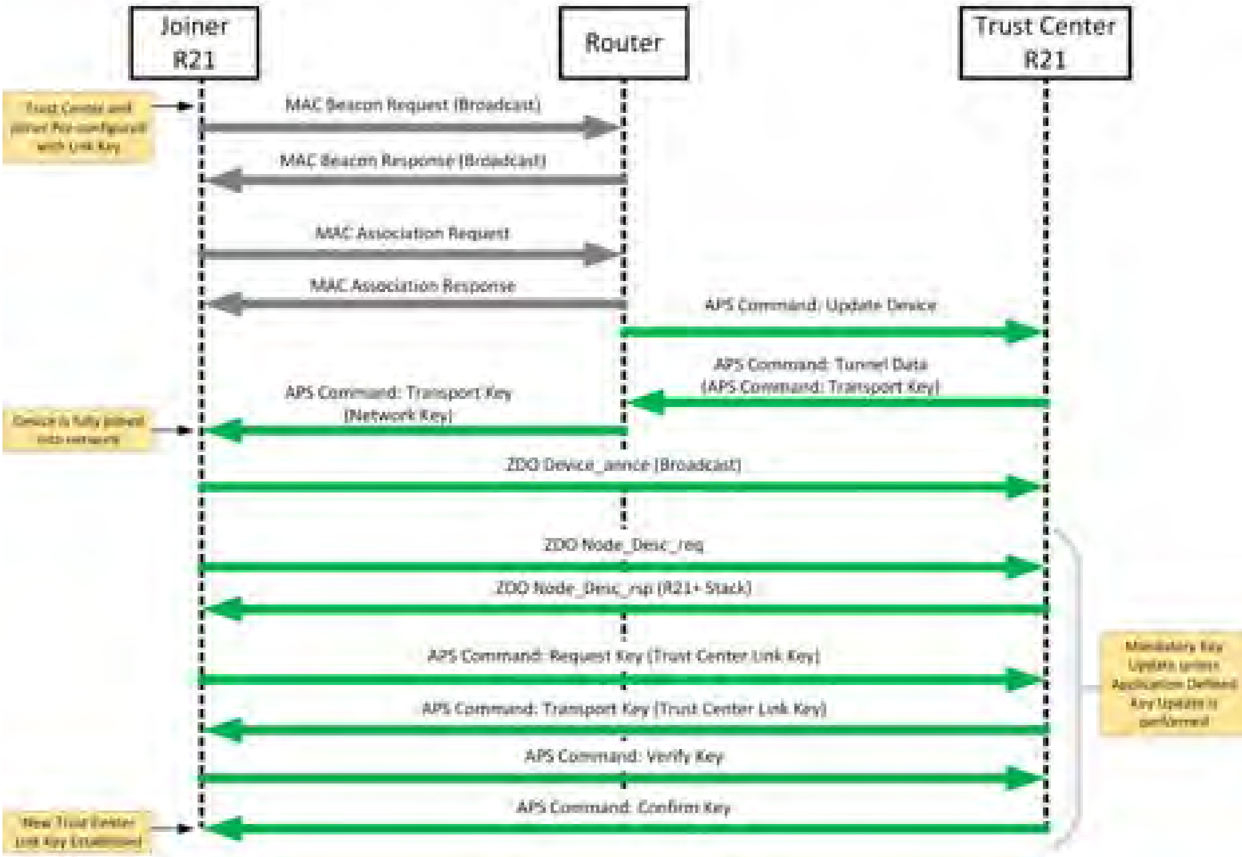


Figure 1-3. Joining in Revision 21

1.1.5.3 Joining in a distributed security network

With Revision 21 a lightweight distributed security model was added that allowed for a network to operate without a trust center such that every router can choose to allow a device on the network. This joining flow is shown in Figure 1-4.

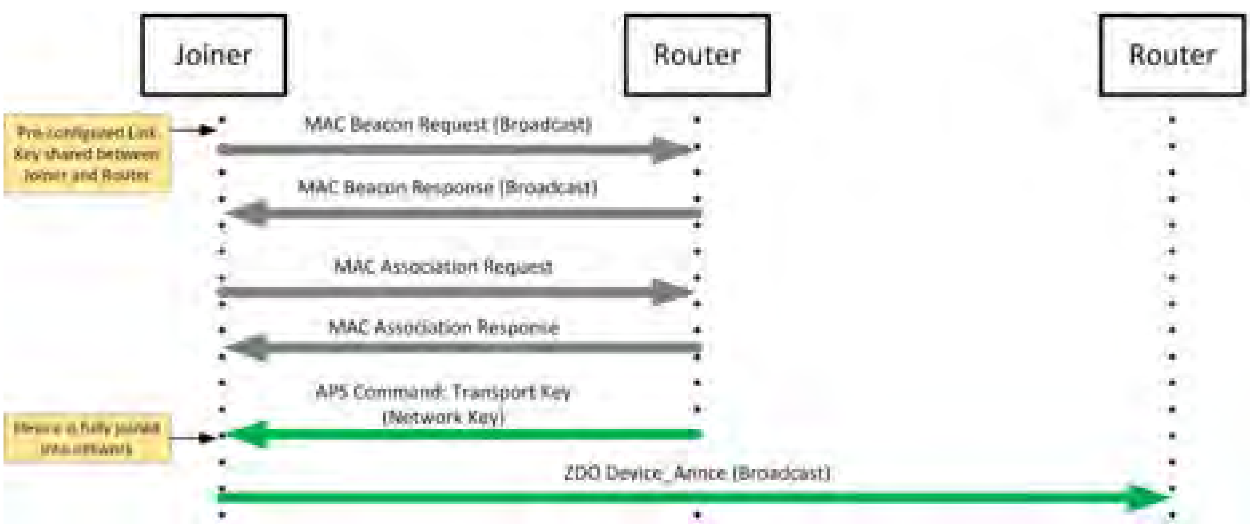


Figure 1-4. Joining a Distributed Network in Revision 21

In distributed networks no Link Key Update is performed.

1.1.5.4 Dynamic Key negotiation before joining

Revision 23 of this specification introduces a mechanism to negotiate a dynamic link key before the device joins the network and receives the network key. This mechanism utilizes Elliptic Curve Diffie-Hellman Ephemeral (ECDHE) or Simple Password Exponential Key Exchange (SPEKE) as the basis for negotiating a key. The negotiation MAY be done anonymously with a well-known passphrase, where the Trust Center has no prior knowledge of the device joining the network, or it MAY be done by using install codes or a secret passphrase to authenticate both sides during key negotiation. This scenario requires that all involved devices (including the parent router) are Revision 23 or later. This is shown in Figure 1-5.

After joining, the device is not required to replace its link key immediately. Instead, it acquires a token that it can use to perform authenticated re-negotiation of its link key in the future.

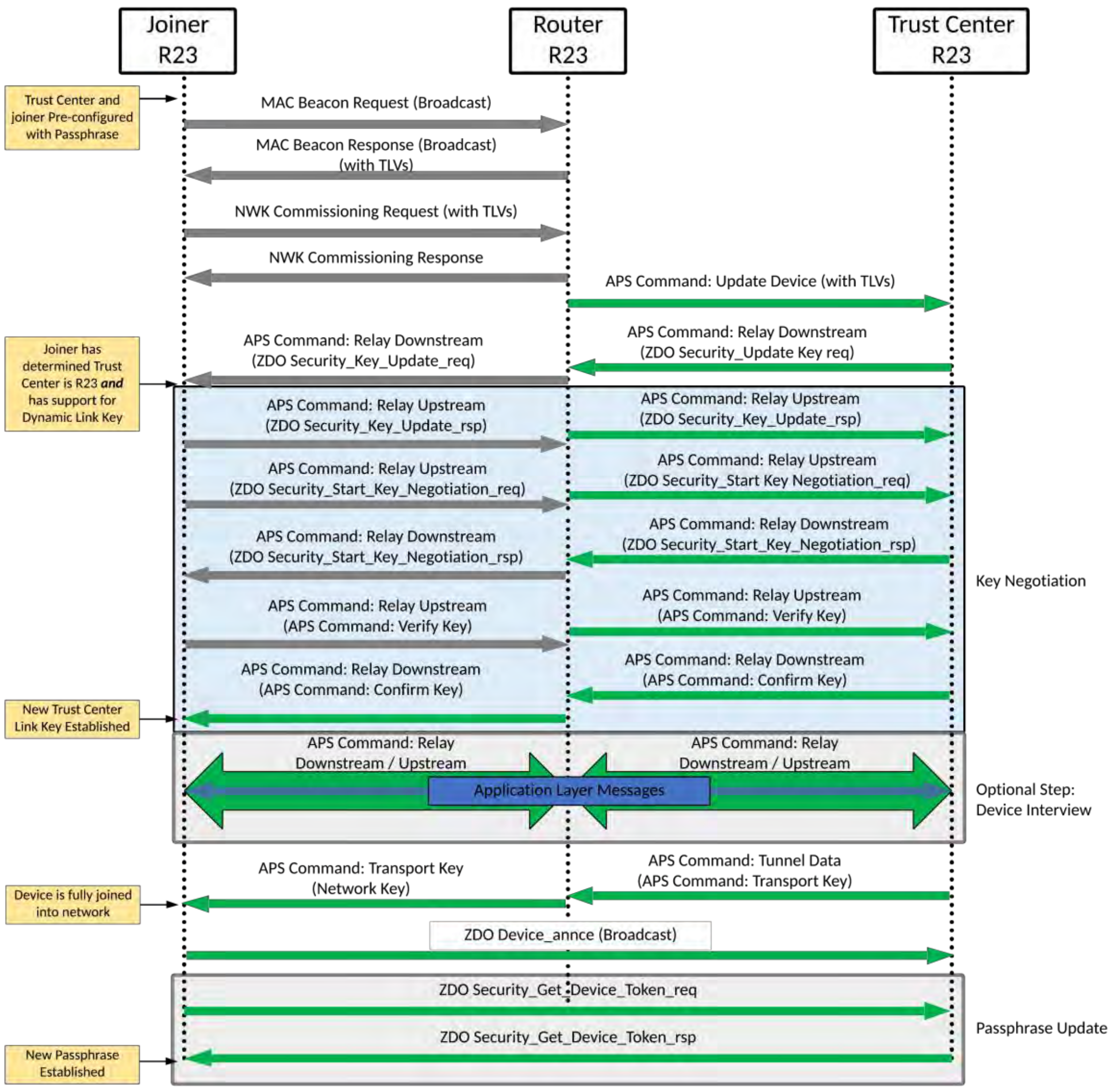


Figure 1-5. Joining in Revision 23 with Dynamic Key Negotiation before Receiving the Network Key

1.1.5.5 Optional Step: Device Interview

Once network commissioning has begun and a dynamic link key has been established the Trust Center Application can opt to query the joining device or an application endpoint with APSData requests encrypted with the Dynamic Link Key prior to authorizing the device on the network.. This period of message exchange is known as the Device

Interview. During this process the joining timeout will be extended until either the TC application allows the device to come on to network by sending the Network Key via Transport Key Mechanism.

1.1.5.6 Dynamic Key negotiation after joining

A device that supports Dynamic Key Negotiation before joining might still join using Static Link Key Joining. This could occur because the parent router is not R23 (and cannot relay key negotiation frames) or the Trust Center does not support Key Negotiation.

After initially joining the network with a static link key the device is required to update that key. If both Trust Center and the device support Dynamic Key Negotiation this SHALL be used as the mechanism to update a link key after the device has joined the network. This is shown in Figure 1-6.

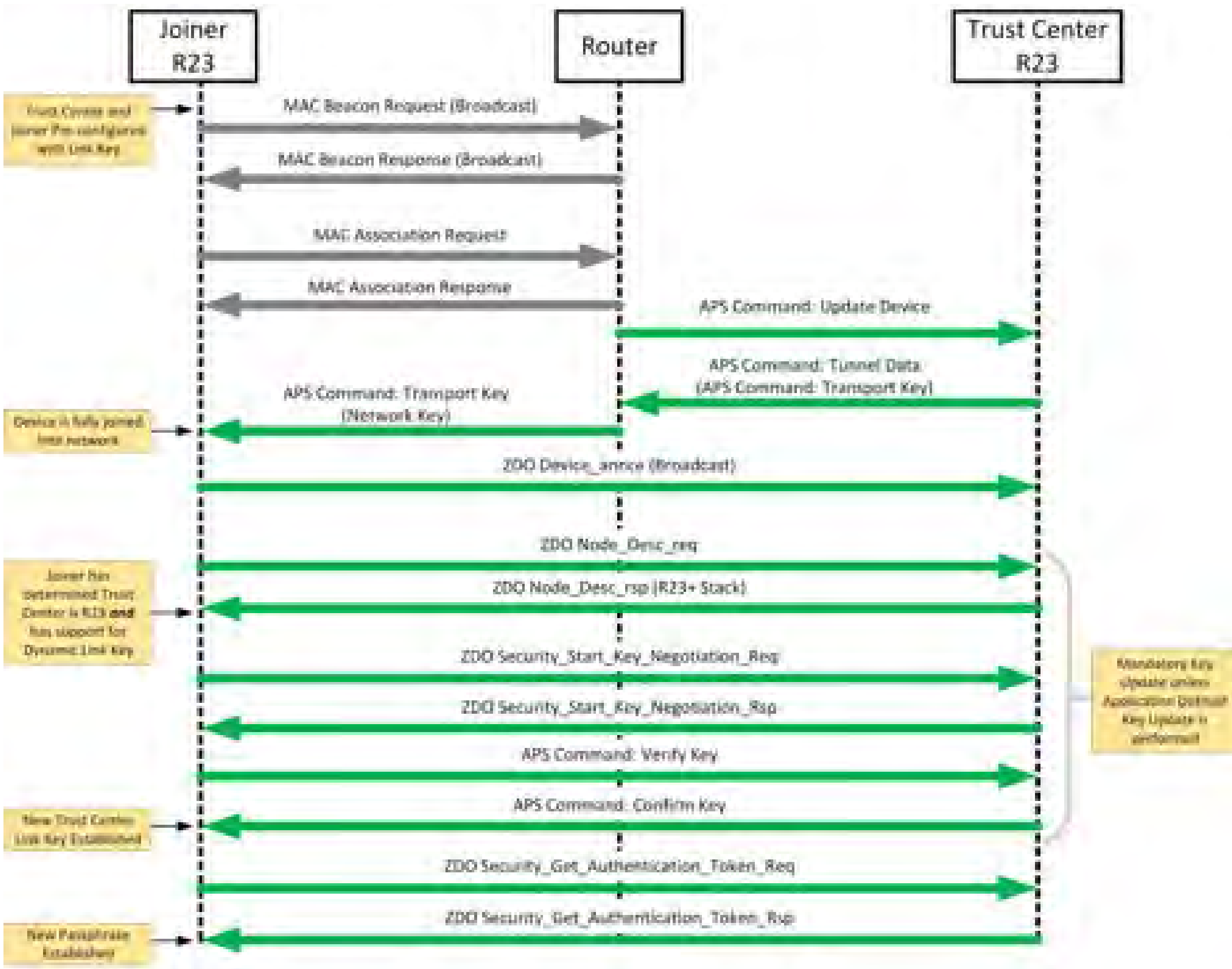


Figure 1-6. Joining in Revision 23 with Dynamic Key Negotiation after Receiving the Network Key

1.1.5.7 Summary of Joining and Link Key Update Mechanisms

Table 1-1 summarizes the Join mechanism and the subsequent link key update after joining for Centralized networks.

Table 1-1. Summary of Join and Key Update Mechanisms for Centralized Networks

Specification Revision	Join Mechanisms	Post Joining Link Key Update
20 and earlier	Static Key Joining	None
	Static Key Joining	Application Defined
21	Static Key Joining	Key Assignment
	Static Key Joining	Application Defined
22	Static Key Joining	Key Assignment
	Static Key Joining	Application Defined
23	Static Key Joining	Key Assignment
	Static Key Joining	Application Defined
	Static Key Joining	Dynamic Key Negotiation Update
	Dynamic Key Negotiation Joining	Dynamic Key Negotiation Update*
	Dynamic Key Negotiation Joining	Application Defined

* Note: A Link Key Update is not required immediately after joining if Dynamic Key Negotiation was used. However, devices can update their link key later using Dynamic Key Negotiation.

Table 1-2 summarizes the Join mechanisms and subsequent link key update after joining.

Table 1-2. Summary of Join and Key Update Mechanisms for Distributed Networks

Specification Revision	Join Mechanisms	Post Joining Link Key Update
All	Static Key Joining	None

1.2 Conventions and Abbreviations

1.2.1 Symbols and Notation

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

1.2.2 Integers, Octets, and Their Representation

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

1.2.3 Transmission Order

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

1. Frame formats are depicted in the order in which they are transmitted by the PHY layer—from left to right—where the leftmost bit is transmitted first in time.

2. Bits within each field are numbered from 0 (leftmost, and least significant) to k-1 (rightmost, and most significant), where the length of the field is k bits.
3. Fields that are longer than a single octet are sent to the PHY in order from the octet containing the lowest numbered bits to the octet containing the highest- numbered bits.

1.2.4 Strings and String Operations

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

1.2.5 Handling Malformed Zigbee and IEEE Std 802.15.4 Frames

If Zigbee messages are received that have mandatory fields missing, the entire message SHALL be ignored. This includes the MAC layer, NWK layer, APS layer, and ZDO layers. The handling of malformed higher layer messages is up to the application layer.

If NWK Commands, APS Commands, or ZDO frames are received that have additional fields over those expected, the expected parts of the field SHALL be processed and the additional fields ignored.

1.2.6 Type Length Value (TLV) Data

Revision 23 of this specification has introduced Type Length Value formatted fields that are appended as new fields in existing commands or present in new commands.

Commands introduced before R23 have a byte packed format that SHOULD NOT be changed due to interoperability considerations. However, it is EXPECTED that those existing commands can be extended without impacting that. Any extensions to existing commands or introduction of new ones will use TLVs.

All Zigbee messages MAY be extended in a future specification. Thus, devices SHALL NOT discard frames when they have additional data beyond defined fields.

1.3 Acronyms

The acronyms used this specification are included in Table 1-3.

Table 1-3. Acronyms Used in this Specification

Acronym	Definition
AIB	Application support sub-layer information base
AF	Application framework
APDU	Application support sub-layer protocol data unit
APL	Application layer
APS	Application support sub-layer
APSDE	Application support sub-layer data entity

Acronym	Definition
APSDE-SAP	Application support sub-layer data entity – service access point
APSME	Application support sub-layer management entity
APSME-SAP	Application support sub-layer management entity – service access point
ASDU	APS service data unit
BRT	Broadcast retry timer
BT	(Filter) Bandwidth (Symbol) Time Product
BTR	Broadcast transaction record
BTT	Broadcast transaction table
CCM*	Counter with CBC-MAC, a cryptographic block cipher mode
CSMA-CA	Carrier sense multiple access – collision avoidance.
CRC	Cyclic Redundancy Check
ED	Energy Detection
ECDHE	Elliptic Curve Diffie-Helman Ephemeral
EPID	Extended PAN ID
FCS	Frame Check Sequence
FEC	Forward Error Correction
FFD	Full function device
FSK	Frequency Shift Keying
GB	Great Britain
GHz	Gigahertz
GPD	Green Power Device
GPDF	Green Power Device Frame
GPEP	Green Power Endpoint
HDR	Header
IB	Information base
IE	Information Element
IEEE	Institute of Electrical and Electronics Engineers
kHz	Kilohertz
LBT	Listen Before Talk. ETSI defined channel access mechanism
LQI	Link quality indicator
LR-WPAN	Low rate wireless personal area network
MAC	Medium access control
MCPS-SAP	Medium access control common part sub-layer service access point

Acronym	Definition
MHz	Megahertz
MIC	Message integrity code
MLME-SAP	Medium access control sub-layer management entity service access point
MSC	Message sequence chart
MSDU	Medium access control sub-layer service data unit
MSG	Message service type
MTU	Maximum Transmission Unit
NBDT	Network broadcast delivery time
NHLE	Next higher layer entity
NIB	Network layer information base
NLDE	Network layer data entity
NLDE-SAP	Network layer data entity – service access point
NLME	Network layer management entity
NLME-SAP	Network layer management entity – service access point
NPDU	Network layer protocol data unit
NSDU	Network service data unit
NWK	Network
OSI	Open systems interconnection
PAN	Personal area network
PD-SAP	Physical layer data service access point
PDU	Protocol data unit
PHR	PHY Header
PHY	Physical layer
PIB	Personal area network information base
PLME-SAP	Physical layer management entity – service access point
POS	Personal operating space
PPDU	PHY Protocol Data Unit
PSDU	PHY Service Data Unit
QOS	Quality of service
RFD	Reduced function device
RREP	Route reply
RREQ	Route request
SAP	Service access point

Acronym	Definition
SFD	Start of Frame Delimiter
SHR	Synchronization Header
SKG	Secret key generation
SSP	Security services provider
SSS	Security services specification
TRD	Technical Requirements Document
WPAN	Wireless personal area network
ZB	Zigbee
ZDO	Zigbee device object

1.4 Glossary

1.4.1 Conformance Language

The key words in Table 1-4 are capitalized in this specification.

Table 1-4. Key Words Used in this Specification

Key Word	Description
EXPECTED	Used to describe the behavior *assumed* by this specification. Other behaviors may also be implemented.
MAY	Indicates flexibility of choice with *no implied preference*.
NOT	Describes that the requirement is the inverse of the behavior specified (that is, SHALL NOT, MAY NOT, etc.)
SHALL	Indicates a mandatory requirement. Designers are required to implement all such mandatory requirements.
SHOULD	Indicates flexibility of choice with a strongly preferred alternative. Equivalent to the phrase *is recommended*.

1.4.2 Conformance Requirements

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

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

Zigbee Protocol Version: The name of the Zigbee protocol version governed by this specification. The protocol version sub-field of the frame control field in the NWK header of all Zigbee Protocol Stack frames conforming to this specification SHALL have a value of 0x02 for all Zigbee frames, and a value of 0x03 for the Zigbee Green Power frames. The protocol version support required by various Zigbee specification revisions appears in Table 1-5.

Table 1-5. Zigbee Protocol Versions

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

A Zigbee device that conforms to this version of the specification may elect to provide backward compatibility with the 2004 Revision of the specification. If it so elects, it SHALL do so by supporting, in addition to the frame formats and features described in this specification version, all frame formats and features as specified in the older version. (All devices in an operating network, regardless of which revisions of the Zigbee specification they support internally, shall, with respect to their external, observable behavior, consistently conform to a single Zigbee protocol version.) A single Zigbee network SHALL NOT contain devices that conform, in terms of their external behavior, to multiple Zigbee protocol versions. [The protocol version of the network to join SHALL be determined by a backwardly compatible device in examining the beacon payload prior to deciding to join the network; or SHALL be established by the application if the device is a Zigbee coordinator.] A Zigbee device conforming to this specification may elect to support only protocol version 0x02, whereby it SHALL join only networks that advertise commensurate beacon payload support. A Zigbee device that conforms to this specification SHALL discard all frames carrying a protocol version sub-field value other than 0x01, 0x02, or0x03. It SHALL process only protocol versions of 0x01 or 0x02, consistent with the protocol version of the network that the device participates within. A Zigbee device that conforms to this specification SHALL pass the frames carrying the protocol version sub-field value 0x03 to the Interpan APS (see Annex G), if it supports the Zigbee Green Power, otherwise it SHALL drop them.

1.4.3 Zigbee Definitions

For the purposes of this standard, the following terms and definitions apply. Terms not defined in this section can be found in [B1].

Access control list: This is a table used by a device to determine which devices are authorized to perform a specific function. This table MAY also store the security material (for example, cryptographic keys, frame counts, key counts, security level information) used for securely communicating with other devices.

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

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

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

Application key: This is a link key transported by the Trust center to a device for the purpose of securing end-to-end communication.

Application object: This is a component of the top portion of the application layer defined by the manufacturer that actually implements the application.

Application profile: This is a collection of device descriptions, which together form a cooperative application. For instance, a thermostat on one node communicates with a furnace on another node. Together, they cooperatively form a heating application profile.

Application support sub-layer protocol data unit: This is a unit of data that is exchanged between the application support sub-layers of two peer entities.

APS command frame: This is a command frame from the APSME on a device addressed to the peer entity on another device.

Association: This is the service provided by the IEEE Std 802.15.4 MAC sub-layer that is used to establish membership in a network.

Attribute: This is a data entity which represents a physical quantity or state. This data is communicated to other devices using commands.

Binding: This is the creation of a unidirectional logical link between a source endpoint/cluster identifier pair and a destination endpoint, which MAY exist on one or more devices.

Broadcast: This is the transmission of a message to every device in a particular PAN belonging to one of a small number of statically defined broadcast groups, for example all routers, and within a given transmission radius measured in hops.

Broadcast jitter: This is a random delay time introduced by a device before relaying a broadcast transaction.

Broadcast transaction record: This is a local receipt of a broadcast message that was either initiated or relayed by a device.

Broadcast transaction table: This is a collection of broadcast transaction records.

Cluster: This is an application message, which MAY be a container for one or more attributes. As an example, the Zigbee Device Profile defines commands and responses. These are contained in Clusters with the cluster identifiers enumerated for each command and response. Each Zigbee Device Profile message is then defined as a cluster. Alternatively, an application profile MAY create sub-types within the cluster known as attributes. In this case, the cluster is a collection of attributes specified to accompany a specific cluster identifier (sub-type messages.)

Cluster identifier: This is a reference to an enumeration of clusters within a specific application profile or collection of application profiles. The cluster identifier is a 16-bit number unique within the scope of each application profile and identifies a specific cluster. Conventions MAY be established across application profiles for common definitions of cluster identifiers whereby each application profile defines a set of cluster identifiers identically. Cluster identifiers are designated as inputs or outputs in the simple descriptor for use in creating a binding table.

Coordinator: This is an IEEE Std 802.15.4 device responsible for associating and disassociating devices into its PAN. A coordinator SHALL be a full-function device (FFD).

Data integrity: This is assurance that the data has not been modified from its original form.

Data key: This is a key derived from a link key used to protect data messages.

Device: This is any entity that contains an implementation of the Zigbee protocol stack.

Device application: This is a special application that is responsible for Device operation. The device application resides on endpoint 0 by convention and contains logic to manage the device's networking and general maintenance features. Endpoints 241-254 are reserved for use by the Device application or common application function agreed within the Connectivity Standards Alliance.

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

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

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

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

Endpoint: This is a particular component within a unit. Each Zigbee device MAY support up to 254 such components.

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

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

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

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

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

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

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

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

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

Maximum Transmission Unit (MTU): The maximum size message that can be sent according to the layer it is being generated at.

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

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

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

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

Neighbor table: This is a table used by a Zigbee device to keep track of other devices within the POS.

Network address: This is the address assigned to a device by the network layer and used by the network layer for routing messages between devices.

Network broadcast delivery time: This is the time required by a broadcast transaction to reach every device of a given network.

Network manager: This is a Zigbee device that implements network management functions as described in Chapter 3, including PAN ID conflict resolution and frequency agility measures in the face of interference.

Network protocol data unit: This is a unit of data that is exchanged between the network layers of two peer entities.

Network service data unit: This is the information that is delivered as a unit through a network service access point.

Node: This is a collection of independent device descriptions and applications residing in a single unit and sharing a common 802.15.4 radio.

Normal operating state: This is the processing which occurs after all startup and initialization processing has occurred and prior to initiation of shutdown processing.

NULL: a parameter or variable value that means unspecified, undefined, or unknown. The exact value of NULL is implementation-specific, and SHALL NOT conflict with any other parameters or values.

Octet: eight bits of data, used as a synonym for a byte.

OctetDuration: transmission time (in seconds) of an octet on PHY layer. This time is calculated as 8/phyBitRate where phyBitRate can be found in Table 1 of [B1]. To get milliseconds from N OctetDurations for 2.4 GHz the follow formula has to be used: N*(8/250000)*1000 where 250000 bit rate on 2.4 GHz and 8 number of bits in one octet.

One-way function: a function whose forward computation is much easier to perform than its inverse.

Orphaned device: a device, typically a Zigbee end device that has lost communication with the Zigbee device through which it has its PAN membership.

PAN coordinator: the principal controller of an IEEE Std 802.15.4-based network that is responsible for network formation. The PAN coordinator SHALL be a full function device (FFD).

PAN information base: a collection of variables in the IEEE Std 802.15.4 standard that are passed between layers, in order to exchange information. This database MAY include the access control list, which stores the security material.

Passphrase: A symmetric secret used as part of a key negotiation protocol.

Personal operating space: the area within reception range of a single device.

Private method: attributes and commands which are accessible to Zigbee device objects only and unavailable to the end applications.

Protocol data unit: the unit of data that is exchanged between two peer entities.

Public method: attributes and commands which are accessible to end applications.

Radio: the IEEE Std 802.15.4 radio that is part of every Zigbee device.

Remote device: the target of a ZDP command.

Route discovery: an operation in which a Zigbee coordinator or Zigbee router attempts to discover a route to a remote device by issuing a route request command frame.

Route discovery table: a table used by a Zigbee coordinator or Zigbee router to store temporary information used during route discovery.

Route reply: a Zigbee network layer command frame used to reply to route requests.

Route request: a Zigbee network layer command frame used to discover paths through the network over which subsequent messages MAY be delivered.

Routing table: a table in which a Zigbee coordinator or Zigbee router stores information required to participate in the routing of frames.

Security token: A term for a generic security item that is given to a device and to verify the identity of a device when negotiating a security key.

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

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

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

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

Zigbee coordinator: an IEEE Std 802.15.4 PAN coordinator.

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

Zigbee end device: an IEEE Std 802.15.4 RFD or FFD participating in a Zigbee network, which is neither the Zigbee coordinator nor a Zigbee router.

Zigbee router: an IEEE Std 802.15.4 FFD participating in a Zigbee network, which is not the Zigbee coordinator but MAY act as an IEEE Std 802.15.4 coordinator within its personal operating space, that is capable of routing messages between devices and supporting associations.

Zigbee 2.4 GHz Coordinator: An IEEE Std 802.15.4-2020 PAN coordinator operating in a Zigbee 2.4 GHz network.

Zigbee 2.4 GHz End Device: An IEEE Std 802.15.4-2020 RFD participating in a Zigbee 2.4 GHz network, which is neither the Zigbee coordinator nor a Zigbee router.

Zigbee 2.4 GHz Router: An IEEE Std 802.15.4-2020 FFD participating in a Zigbee 2.4 GHz network, which is not the Zigbee coordinator but MAY act as an IEEE Std 802.15.4-2020 coordinator within its personal operating space, that is capable of routing messages between devices and supporting associations

Zigbee Sub-GHz Router: An IEEE Std 802.15.4-2020 FFD participating in a Zigbee Sub- GHz network, which is not the Zigbee coordinator but MAY act as an IEEE Std 802.15.4-2020 coordinator within its personal operating space, that is capable of routing messages between devices and supporting associations. Zigbee Sub-GHz Router (ZSR) is supported in R22 with power control on end device to routers and end devices to coordinators links. There is no power control for router to router, and router to coordinator links.

Zigbee Multi-MAC Selection Router: An IEEE Std 802.15.4-2020 FFD participating in a Zigbee Sub-GHz **or** 2.4 GHz network but **not** in both bands. Power control only on Sub-GHz interface and not on the 2.4 GHz interface. Router in Sub-GHz mode in R22 will support power control on end device to routers and end devices to coordinators links. There is no power control for router to router, and router to coordinator links.

Zigbee Multi-MAC Switch Router: An IEEE Std 802.15.4-2020 FFD participating in a Zigbee Sub-GHz **and** 2.4 GHz network. In R22 only allows a single Zigbee Multi-MAC Switch Router in the network integrated into the Zigbee Multi-MAC Switch Coordinator

Zigbee Multi-MAC Switch Coordinator: An IEEE Std 802.15.4-2020 PAN coordinator operating in a Zigbee 2.4 GHz network **and** in Sub-GHz band.

Zigbee Multi-MAC Selection End Device: An IEEE Std 802.15.4-2020 RFD participating in a Zigbee 2.4 GHz network **or** the Sub-GHz network which is neither the Zigbee coordinator nor a Zigbee router.

Zigbee Sub-GHz End Device: An IEEE Std 802.15.4-2020 RFD participating in a Zigbee Sub-GHz network which is neither the Zigbee coordinator nor a Zigbee router.

1.5 References

The following standards contain provisions, which, through reference in this document, constitute provisions of this standard. Normative references are given in and and informative references are given in At the time of publication, the editions indicated were valid. All standards are subject to Revision, and parties to agreements based on this standard are encouraged to investigate the possibility of applying the most recent editions of the references, as indicated in this section.

1.5.1 Zigbee/IEEE References

[B1] 802.15.4-2020, IEEE Standard for Local and metropolitan area networks--Part 15.4: Low-Rate Wireless Personal Area Networks (LR-WPANs)

[B2] Document 20-27688-032: Zigbee Direct specification

[B3] Document 03-285r00: Suggestions for the Improvement of the IEEE 802.15.4 Standard, July 2003.

[B4] Document 09-5499r26: Green Power specification

[B5] Document 14-0563-16: Zigbee PRO Green Power feature specification Basic functionality set

1.5.2 Normative References

[B6] ISO/IEC 646:199 Information technology — ISO 7-bit coded character set for information interchange.

[B7] ANSI X9.63-2001, Public Key Cryptography for the Financial Services Industry - Key Agreement and Key Transport Using Elliptic Curve Cryptography, American Bankers Association, November 20, 2001. Available from http://www.ansi.org.

[B8] FIPS Pub 197, Advanced Encryption Standard (AES), Federal Information Processing Standards Publication 197, US Department of Commerce/N.I.S.T, Springfield, Virginia, November 26, 2001. Available from http://csrc.nist.gov/.

[B9] FIPS Pub 198, The Keyed-Hash Message Authentication Code (HMAC), Federal Information Processing Standards Publication 198, US Department of Commerce/N.I.S.T., Springfield, Virginia, March 6, 2002. Available from http://csrc.nist.gov/.

[B10] NIST Pub 800-38A 2001 ED, Recommendation for Block Cipher Modes of Operation — Methods and Techniques, NIST Special Publication 800-38A, 2001 Edition, US Department of Commerce/N.I.S.T., December 2001. Available from http://csrc.nist.gov/.

[B11] NIST, Random Number Generation and Testing. Available from http://csrc.nist.gov/rng/.

[B12] [EN 300-220] ETSI EN 300 220-1 V2.4.1 (2012-01) Electromagnetic compatibility and Radio spectrum Matters (ERM); Short Range Devices (SRD); Radio equipment to be used in the 25 MHz to 1000 MHz frequency range with power levels ranging up to 500 mW; Part 1: Technical characteristics and test methods

[B13] [EN 300-220] ETSI EN 300 220-1 V3.1.1 (2017-07) – Draft / unpublished) Electromagnetic compatibility and Radio spectrum Matters (ERM); Short Range Devices (SRD); Radio equipment to be used in the 25 MHz to 1000 MHz frequency range with power levels ranging up to 500 mW; Part 1: Technical characteristics and test methods

[B14] [EN 303-204] ETSI EN 303 204-1 V1.1.0 (2014-06)Electromagnetic compatibility and Radio spectrum Matters (ERM); Network Based Short Range Devices (SRD); Radio equipment to be used in the 870 MHz to 876 MHz frequency range with power levels ranging up to 500 mW; Part 1:Technical characteristics and test methods

1.5.3 Informative References

[B15] FIPS140- 2 ISO/IEC 7498-1:1994 Information technology — Open systems interconnection — Basic reference model: The basic model.

[B16] ISO/IEC 9646-1:1991, Information technology — Open systems Interconnection — Conformance testing methodology and framework — Part 1: General concepts.

[B17] ISO/IEC 9646-7:1995, Information technology — Open Systems Interconnection — Conformance testing methodology and framework — Part 7. Implementation conformance statements.

[B18] A.J. Menezes, P.C. van Oorschot, S.A. Vanstone, Handbook of Applied Cryptography, Boca Raton: CRC Press, 1997.

[B19] FIPS Pub 113, Computer Data Authentication, Federal Information Processing Standard Publication 113, US Department of Commerce/N.I.S.T., May 30, 1985. Available from http://csrc.nist.gov/.

[B20] A. Langley, M. Hamburg, S. Turner, “Elliptic Curves for Security”, RFC 7748, Internet Engineering Task Force (IETF), 2016

[B21] ISO/IEC 11770-4:2017: Information technology — Security techniques — Key management — Part 4: Mechanisms based on weak secrets, 2017

[B22] 1363.2-2008 - IEEE Standard Specification for Password-Based Public-Key Cryptographic Techniques, 2008

[B23] F. Hao, and S. F. Shahandashti, “The SPEKE Protocol Revisited”. In: L. Chen, C. Mitchell (eds), “Security Standardisation Research – SSR 2014”, Lecture Notes in Computer Science, vol 8893, Springer, Berlin, Heidelberg, 2014.

[B24] F. Hao, R. Metere, S. F. Shahandashti and C. Dong, "Analyzing and Patching SPEKE in ISO/IEC," in IEEE Transactions on Information Forensics and Security, vol. 13, no. 11, pp. 2844-2855, 2018

CHAPTER 2. APPLICATION LAYER SPECIFICATION

2.1 General Description

The Zigbee stack architecture includes a number of layered components including the IEEE Std 802.15.4 Medium Access Control (MAC) layer, Physical (PHY) layer, and the Zigbee Network (NWK) layer. Each component provides an application with its own set of services and capabilities. Although this chapter may refer to other components within the Zigbee stack architecture, its primary purpose is to describe the component labeled Application (APL) Layer shown in Figure 1-1.

As shown in Figure 1-1, the Zigbee application layer consists of the APS sub-layer, the ZDO (containing the ZDO management plane), and the manufacturer-defined application objects.

2.1.1 Application Support Sub-Layer (APS)

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

1. The APS data entity (APSDE) through the APSDE service access point (APSDE-SAP).
2. The APS management entity (APSME) through the APSME service access point (APSME-SAP).

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

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

2.1.2 Application Framework

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

Up to 254 distinct application objects can be defined, each identified by an endpoint from 1 to 254. Two additional endpoints are defined for APSDE-SAP usage: endpoint 0 is reserved for the data interface to the Zigbee Device Object (ZDO) and endpoint 255 is reserved for the data interface function to broadcast data to all application objects. Endpoints 241-254 are assigned by the Connectivity Standards Alliance and SHALL NOT be used without approval. The Green Power cluster, if implemented, SHALL use endpoint 242.

2.1.2.1 Application Profiles

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

2.1.2.2 Clusters

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

2.1.3 Zigbee Device Objects

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

1. Initializing the APS, NWK, and the Security Service Provider.
2. Assembling configuration information from the end applications to determine and implement discovery, security management, network management, and binding management.

The ZDO presents public interfaces to the application objects in the application framework layer for control of device and network functions by the application objects. The ZDO interfaces with the lower portions of the Zigbee protocol stack, on endpoint 0, through the APSDE-SAP for data, and through the APSME-SAP and NLME-SAP for control messages. The public interface provides address management of the device, discovery, binding, and security functions within the application framework layer of the Zigbee protocol stack. The ZDO is fully described in section 2.5.

2.1.3.1 Device Discovery

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

2.1.3.2 Service Discovery

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

2.2 Zigbee Application Support Sub-Layer

2.2.1 Scope

This section specifies the portion of the application layer providing the service specification and interface to both the manufacturer-defined application objects and the Zigbee device objects. The specification defines a data service to allow the application objects to transport data, and a management service providing mechanisms for binding. In addition, it also defines the application support sub-layer frame format and frame-type specifications.

2.2.2 Purpose

The purpose of this section is to define the functionality of the Zigbee APS. This functionality is based on both the driver functionality necessary to enable correct operation of the Zigbee network layer and the functionality required by the manufacturer-defined application objects.

2.2.3 Application Support Sub-Layer Overview

The application support sub-layer provides the interface between the network layer and the application layer through a general set of services for use by both the ZDO and the manufacturer-defined application objects. These services are offered via two entities: the data service and the management service. The APS data entity (APSDE) provides the data transmission service via its associated SAP, the APSDE-SAP. The APS management entity (APSME) provides the management service via its associated SAP, the APSME-SAP, and maintains a database of managed objects known as the AIB.

2.2.3.1 Application Support Sub-Layer Data Entity (APSDE)

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

The APSDE will provide the following services:

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

2.2.3.2 Application Support Sub-Layer Management Entity (APSME)

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

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

In addition, the APSME will provide the following services:

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

2.2.4 Service Specification

The APS sub-layer provides an interface between a next higher layer entity (NHLE) and the NWK layer. The APS sub-layer conceptually includes a management entity called the APS sub-layer management entity (APSME). This entity provides the service interfaces through which sub-layer management functions MAY be invoked. The APSME is also responsible for maintaining a database of managed objects pertaining to the APS sub-layer. This database is referred to as the APS sub-layer information base (AIB). Figure 2-1 depicts the components and interfaces of the APS sub-layer.

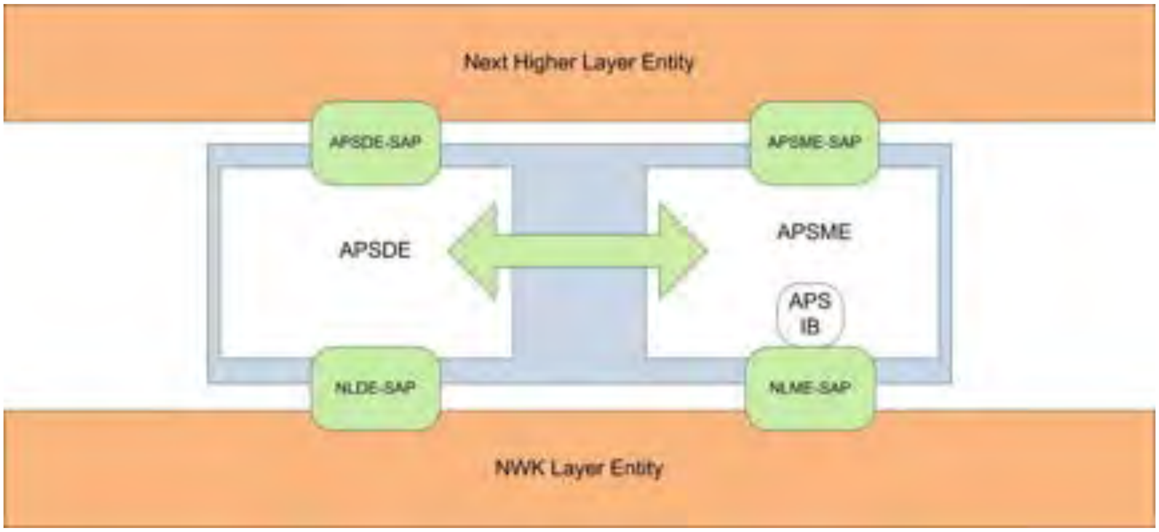


Figure 2-1. The APS Sub-Layer Reference Model

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

2.2.4.1 APS Data Service

The APS sub-layer data entity SAP (APSDE-SAP) supports the transport of application protocol data units between peer application entities. Table 2-1 lists the primitives supported by the APSDE-SAP. Each of these primitives will be discussed in the following sections.

Table 2-1. APSDE-SAP Primitives

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

2.2.4.1.1 APSDE-DATA.request

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

2.2.4.1.1.1 Semantics of the Service Primitive

The semantics of this primitive are as follows:

```
APSDE-DATA.request          {
                            DstAddrMode,
                            DstAddress,
                            DstEndpoint,
                            ProfileId,
                            ClusterId,
                            SrcEndpoint,
                            ASDULength,
                            ASDU,
                            TxOptions,
                            UseAlias,
                            AliasSrcAddr,
                            AliasSeqNumber,
                            RadiusCounter
                            nwkBroadcastAddress
                            }
```

Table 2-2 specifies the parameters for the APSDE-DATA.request primitive. Support of the parameters UseAlias, AliasSrcAddr, and AliasSeqNumb in the APSDE-DATA.request primitive is required if Green Power feature is supported by the implementation.

Table 2-2. APSDE-DATA.request Parameters

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

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

Name	Type	Valid Range	Description
AliasSeqNumb	integer	0x00-0xff	The sequence number to be used for this NSDU. If the *UseAlias* parameter has a value of FALSE, the *AliasSeqNumb* parameter is ignored.
RadiusCounter	Unsigned integer	0x00-0xff	The distance, in hops, that a transmitted frame will be allowed to travel through the network.
nwkBroadcastAddress	16-bit address	0xFFFC - 0xFFFF	This indicates the broadcast address used for multicast messages (DstAddrMode = 0x01).

2.2.4.1.1.2 When Generated

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

2.2.4.1.1.3 Effect on Receipt

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

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

If the DstAddrMode parameter is set to 0x03, the DstAddress parameter contains an extended 64-bit IEEE address and SHALL first be mapped to a corresponding 16-bit NWK address by using the nwkAddressMap attribute of the NIB (see Table 3-46). If a corresponding 16-bit NWK address could not be found, the APSDE issues the APSDE-DATA.confirm primitive with a status of NO_SHORT_ADDRESS. If a corresponding 16-bit NWK address is found, it will be used in the invocation of the NLDE-DATA.request primitive and the value of the DstEndpoint parameter will be placed in the resulting APDU. The delivery mode sub-field of the frame control field of the APS header SHALL have a value of 0x00 in this case.

If the DstAddrMode parameter has a value of 0x01, indicating group addressing, the DstAddress parameter will be interpreted as a 16-bit group address. This address will be placed in the group address field of the APS header, the DstEndpoint parameter will be ignored, and the destination endpoint field will be omitted from the APS header. The delivery mode sub-field of the frame control field of the APS header SHALL have a value of 0x03 in this case. The nwkBroadcastAddress passed to the APSDE-DATA.request primitive SHALL be passed to the NLDE-DATA.request primitive.

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

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

The parameters UseAlias, AliasSrcAddr and AliasSeqNumb SHALL be used in the invocation of the NLDE-DATA.request primitive. If UseAlias is set to TRUE, the AliasSeqNumb value SHALL be copied into the APS Counter field instead of using the device's own value.

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

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

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

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

If the ASDU to be transmitted is large than will fit in a single frame and the destination is a broadcast address then the APSDE SHALL return a status of ASDU_TOO_LONG error via the APSDE-DATA.confirm.

If the ASDU to be transmitted is larger than will fit in a single frame and the destination is not a broadcast address,, then the device SHALL first determine whether the destination supports fragmentation and the size of the maximum incoming transfer size of the destination. This is done by initiating a ZDO Node_Desc_req to the device including the Fragmentation Parameters Global TLV of the local device. The ZDO Node_Desc_rsp will include the support and the maximum buffer size the destination can support. The sender of the message SHALL wait apsZdoResponseTimeout seconds for the response. If no response is received a status of NO_ACK is returned to the application via the APSDE-DATA.confirm.

If the ASDU to be transmitted is larger than will fit in a single frame, an acknowledged transmission is requested, and the fragmentation permitted flag of the TxOptions field is set to 1, and the ASDU is not too large to be handled by the APSDE, then the ASDU SHALL be fragmented across multiple APDUs, as described in section 2.2.8.4.5. Transmission and security processing where requested, SHALL be carried out for each individual APDU independently. Note that fragmentation SHALL NOT be used unless relevant higher-layer interactions explicitly indicate that fragmentation is permitted for the frame being sent, and that the other end is able to receive the fragmented transmission, both in terms of number of blocks and total transmission size.

Figure 2-2 indicates the overall flow of how the APSDE state machine will manage message for fragmentation.

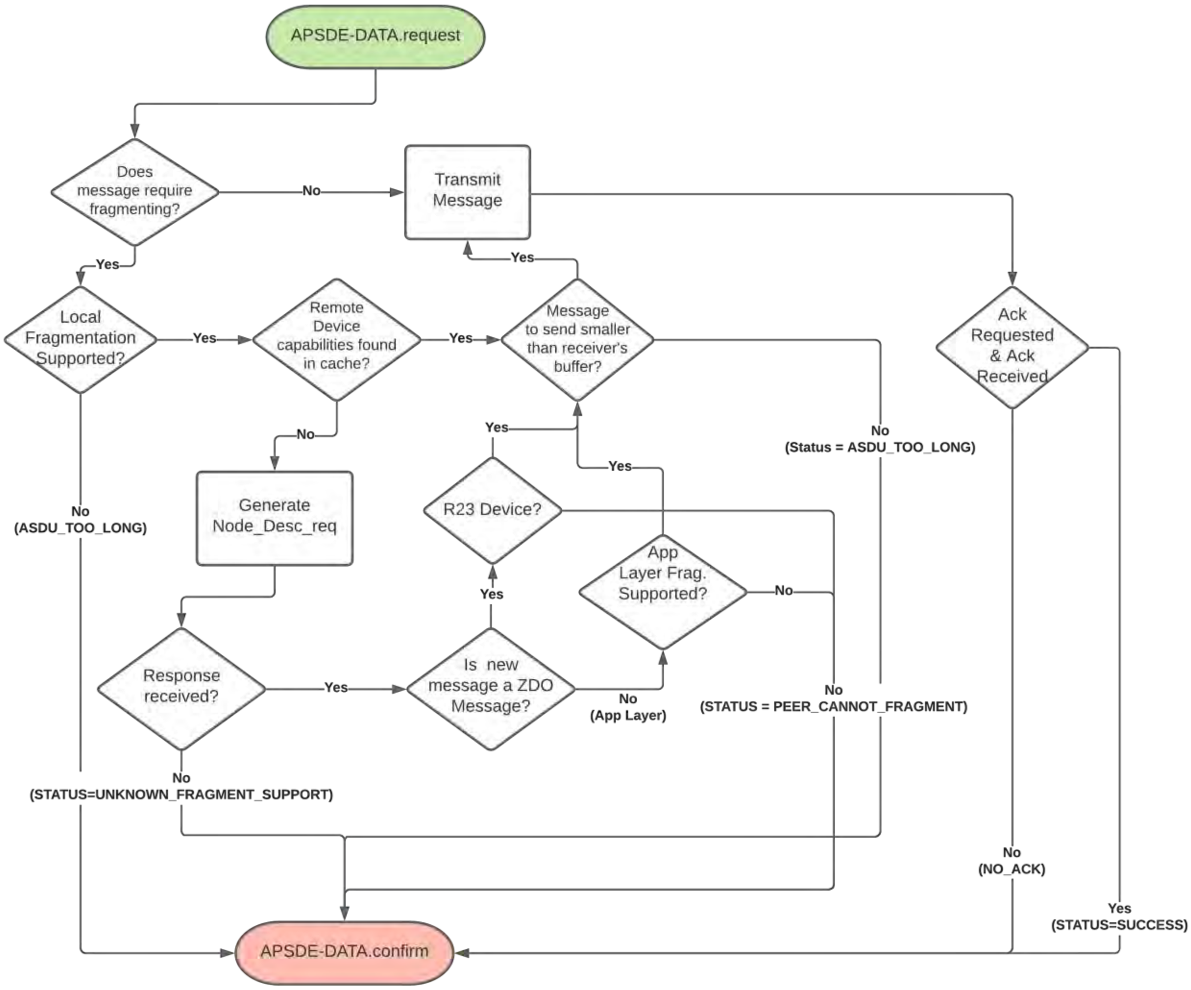


Figure 2-2. APSDE-DATA.request Process Flow

2.2.4.1.2 APSDE-DATA.confirm

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

2.2.4.1.2.1 Semantics of the Service Primitive

This semantics of this primitive are as follows:

```
APSDE-DATA.confirm          {
                            DstAddrMode,
                            DstAddress,
                            DstEndpoint,
                            SrcEndpoint,
                            Status,
                            TxTime
                            }
```

Table 2-3 specifies the parameters for the APSDE-DATA.confirm primitive.

Table 2-3. APSDE-DATA.confirm Parameters

Name	Type	Valid Range	Description
DstAddrMode	Integer	0x00 – 0xff	The addressing mode for the destination address used in this primitive and of the APDU to be transferred. This parameter can take one of the non-reserved values from the following list: 0x00 = DstAddress and DstEndpoint not present 0x01 = 16-bit group address for DstAddress; DstEndpoint not present 0x02 = 16-bit address for DstAddress and DstEndpoint present 0x03= 64-bit extended address for DstAddress and DstEndpoint present 0x04 – 0xff = reserved
DstAddress	Address	As specified by the DstAddrMode parameter	The individual device address or group address of the entity to which the ASDU is being transferred.
DstEndpoint	Integer	0x00 – 0xff	This parameter SHALL be present if, and only if, the DstAddrMode parameter has a value of 0x02 or 0x03 and, if present, shall be the number of the individual endpoint of the entity to which the ASDU is being transferred.
SrcEndpoint	Integer	0x00 – 0xfe	The individual endpoint of the entity from which the ASDU is being transferred.
Status	Enumeration	SUCCESS, NO_SHORT_ADDRESS, NO_BOUND_DEVICE, SECURITY_FAIL, NO_ACK, ASDU_TOO_LONG, PEER_CANNOT_FRAGMENT, UN-KNOWN_FRAGMENT_SUPPORT or any status values returned from the NLDE-DATA.confirm primitive.	The status of the corresponding request.
TxTime	Integer	Implementation specific	A time indication for the transmitted packet based on the local clock, as provided by the NWK layer.

2.2.4.1.2.2 When Generated

This primitive is generated by the local APS sub-layer entity in response to an APSDE-DATA.request primitive. This primitive returns a status of either SUCCESS, indicating that the request to transmit was successful, or an error code of NO_SHORT_ADDRESS, NO_BOUND_DEVICE, SECURITY_FAIL, ASDU_TOO_LONG, or any status values returned from the NLDE-DATA.confirm primitive. The reasons for these status values are fully described in section 2.2.4.1.1.3.

2.2.4.1.2.3 Effect on Receipt

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

2.2.4.1.3 APSDE-DATA.indication

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

2.2.4.1.3.1 Semantics of the Service Primitive

The semantics of this primitive are as follows:

```
APSDE-DATA.indication          {
                               DstAddrMode,
                               DstAddress,
                               DstEndpoint,
                               SrcAddrMode,
                               SrcAddress,
                               SrcEndpoint,
                               ProfileId,
                               ClusterId,
                               asduLength,
                               asdu,
                               Status,
                               SecurityStatus,
                               LinkQuality,
                               KeyIndex,
                               RxTime,
                               DeviceKeyPairEntry
                               }
```

Table 2-4 specifies the parameters for the APSDE-DATA.indication primitive.

Table 2-4. APSDE-DATA.indication Parameters

Name	Type	Valid Range	Description
DstAddrMode	Integer	0x00 - 0xff	The addressing mode for the destination address used in this primitive and of the APDU that has been received. This parameter can take one of the non-reserved values from the following list: 0x00 = reserved 0x01 = 16-bit group address for DstAddress; DstEndpoint not present

Name	Type	Valid Range	Description
			0x02 = 16-bit address for DstAddress and DstEndpoint present 0x03 = 64-bit extended address for DstAddress and DstEndpoint present. 0x04 = 64-bit extended address for DstAddress, but DstEndpoint NOT present. 0x05 – 0xff = reserved
DstAddress	Address	As specified by the DstAddrMode parameter	The individual device address or group address to which the ASDU is directed.
DstEndpoint	Integer	0x00 – 0xfe	The target endpoint on the local entity to which the ASDU is directed.
SrcAddrMode	Integer	0x00 – 0xff	The addressing mode for the source address used in this primitive and of the APDU that has been received. This parameter can take one of the non-reserved values from the following list: 0x00 = reserved 0x01 = reserved 0x02 = 16-bit short address for SrcAddress and SrcEndpoint present 0x03 = 64-bit extended address for SrcAddress and SrcEndpoint present 0x04 = 64-bit extended address for SrcAddress, but SrcEndpoint NOT present. 0x05 – 0xff = reserved
SrcAddress	Address	As specified by the SrcAddrMode parameter	The individual device address of the entity from which the ASDU has been received.
SrcEndpoint	Integer	0x00 – 0xfe	The number of the individual endpoint of the entity from which the ASDU has been received.
ProfileId	Integer	0x0000 – 0xffff	The identifier of the profile from which this frame originated.
ClusterId	Integer	0x0000 – 0xffff	The identifier of the received object.

Name	Type	Valid Range	Description
asduLength	Integer	-	The number of octets comprising the ASDU being indicated by the APSDE.
asdu	Set of octets	-	The set of octets comprising the ASDU being indicated by the APSDE.
Status	Enumeration	SUCCESS, DEFRAG_UNSUPPORTED, DEFRAG_DEFERRED or any status returned from the security processing of the frame	The status of the incoming frame processing.
SecurityStatus	Enumeration	UNSECURED, SECURED_NWK_KEY, or SECURED_LINK_KEY	UNSECURED if the ASDU was received without any security. SECURED_NWK_KEY if the received ASDU was secured with the NWK key. SECURED_LINK_KEY if the ASDU was secured with a link key.
LinkQuality	Integer	0x00 - 0xff	The link quality indication delivered by the NLDE.
KeyIndex	Integer	0 – 255	This value is only valid when SecurityStatus is set to SECURED_LINK_KEY. This indicates the index in the *apsDeviceKeyPairSet* table that was used to decrypt the incoming packet. The application may use this index to obtain all the details about the key via an APSME-GET.request.
RxTime	Integer	Implementation-specific	A time indication for the received packet based on the local clock, as provided by the NWK layer.
DeviceKeyPairEntry	Handle	NULL or pointer	If SecurityStatus indicates SECURED_LINK_KEY this will be a handle to the DeviceKeyPairEntry that was used during APS decryption. Otherwise, it will be NULL. This can be used by the application to determine the security parameters of the link key associated with the sending device and thus apply any application layer policies to the message.

2.2.4.1.3.2 When Generated

This primitive is generated by the APS sub-layer and issued to the next higher layer on receipt of an appropriately addressed data frame from the local NWK layer entity or following receipt of an APSDE-DATA. request in which the DstAddrMode parameter was set to 0x00 and the binding table entry has directed the frame to the device itself. If the frame control field of the ASDU header indicates that the frame is secured, security processing SHALL be done as specified in section 4.4.1.

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

If the frame control field of the APDU header indicates that the frame is secured, then security processing SHALL be undertaken as specified in section 4.4.1. If the security processing fails, the APSDE sets the Status parameter to the security error code returned from the security processing.

If the frame is not secured or the security processing was successful, the APSDE SHALL check for the frame being fragmented. If the extended header is included in the APDU header and the fragmentation sub-field of the extended frame control field indicates that the frame is fragmented but this device does not support fragmentation, the APSDE sets the Status parameter to DEFRAG_UNSUPPORTED. If the extended header is included in the APDU header, the fragmentation sub-field of the extended frame control field indicates that the frame is fragmented and the device supports fragmentation, but is not currently able to defragment the frame, the APSDE sets the Status parameter to DEFRAG_DEFERRED.

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

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

The APS sub-layer entity SHALL attempt to map the source address from the received frame to its corresponding extended 64-bit IEEE address by using the nwkAddressMap attribute of the NIB (see Table 3-46). If a corresponding 64-bit IEEE address was found, the APSDE issues this primitive with the SrcAddrMode parameter set to 0x03 and the SrcAddress parameter set to the corresponding 64-bit IEEE address. If a corresponding 64-bit IEEE address was not found, the APSDE issues this primitive with the SrcAddrMode parameter set to 0x02, and the SrcAddress parameter set to the 16-bit source address as contained in the received frame.

2.2.4.1.3.3 Effect on Receipt

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

2.2.4.2 APS Management Service

The APS management entity SAP (APSME-SAP) supports the transport of management commands between the next higher layer and the APSME. Table 2-5 summarizes the primitives supported by the APSME through the APSME-SAP interface. See the following sections for more details on the individual primitives.

Table 2-5. Summary of the Primitives Accessed Through the APSME-SAP

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

2.2.4.3 Binding Primitives

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

Only a device supporting a binding table MAY process these primitives. If any other device receives these primitives from their next higher layer, the primitives SHOULD be rejected.

2.2.4.3.1 APSME-BIND.request

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

2.2.4.3.1.1 Semantics of the Service Primitive

The semantics of this primitive are as follows:

```
APSME-BIND.request          {
                            SrcAddr,
                            SrcEndpoint,
                            ClusterId,
                            DstAddrMode,
                            DstAddr,
                            DstEndpoint
                            }
```

Table 2-6 specifies the parameters for the APSME-BIND.request primitive.

Table 2-6. APSME-BIND.request Parameters

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

2.2.4.3.1.2 When Generated

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

2.2.4.3.1.3 Effect on Receipt

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

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

2.2.4.3.2 APSME-BIND.confirm

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

2.2.4.3.2.1 Semantics of the Service Primitive

The semantics of this primitive are as follows:

```
APSME-BIND.confirm              {
                                Status,
                                SrcAddr,
                                SrcEndpoint,
                                ClusterId,
                                DstAddrMode,
                                DstAddr,
                                DstEndpoint
                                }
```

Table 2-7 specifies the parameters for the APSME-BIND.confirm primitive.

Table 2-7. APSME-BIND.confirm Parameters

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

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

2.2.4.3.2.2 When Generated

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

2.2.4.3.2.3 Effect on Receipt

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

2.2.4.3.3 APSME-UNBIND.request

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

2.2.4.3.3.1 Semantics of the Service Primitive

The semantics of this primitive are as follows:

```
APSME-UNBIND.request        {
                            SrcAddr,
                            SrcEndpoint,
                            ClusterId,
                            DstAddrMode,
                            DstAddr,
                            DstEndpoint
                            }
```

Table 2-8 specifies the parameters for the APSME-UNBIND.request primitive.

Table 2-8. APSME-UNBIND.request Parameters

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

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

2.2.4.3.3.2 When Generated

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

2.2.4.3.3.3 Effect on Receipt

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

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

2.2.4.3.4 APSME-UNBIND.confirm

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

2.2.4.3.4.1 Semantics of the Service Primitive

The semantics of this primitive are as follows:

```
APSME-UNBIND.confirm          {
                              Status,
                              SrcAddr,
                              SrcEndpoint,
                              ClusterId,
                              DstAddrMode,
                              DstAddr,
                              DstEndpoint
                              }
```

Table 2-9 specifies the parameters for the APSME-UNBIND.confirm primitive.

Table 2-9. APSME-UNBIND.confirm Parameters

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

2.2.4.3.4.2 When Generated

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

2.2.4.3.4.3 Effect on Receipt

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

2.2.4.4 Information Base Maintenance

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

2.2.4.4.1 APSME-GET.request

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

2.2.4.4.1.1 Semantics of the Service Primitive

The semantics of this primitive are as follows:

```
APSME-GET.request          {
                           AIBAttribute
                           }
```

Table 2-10 specifies the parameters for this primitive.

Table 2-10. APSME-GET.request Parameters

Name	Type	Valid Range	Description
AIBAttribute	Integer	See Table 2-24.	The identifier of the AIB attribute to read.

2.2.4.4.1.2 When Generated

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

2.2.4.4.1.3 Effect on Receipt

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

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

2.2.4.4.2 APSME-GET.confirm

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

2.2.4.4.2.1 Semantics of the Service Primitive

The semantics of this primitive are as follows:

```
APSME-GET.confirm            {
                             Status,
                             AIBAttribute,
                             AIBAttributeLength,
                             AIBAttributeValue
                             }
```

Table 2-11 specifies the parameters for this primitive.

Table 2-11. APSME-GET.confirm Parameters

Name	Type	Valid Range	Description
Status	Enumeration	SUCCESS or UNSUPPORTED_ ATTRIBUTE	The results of the request to read an AIB attribute value.
AIBAttribute	Integer	See Table 2-24.	The identifier of the AIB attribute that was read.
AIBAttributeLength	Integer	0x0001 – 0xffff	The length, in octets, of the attribute value being returned.
AIBAttributeValue	Various	Attribute-specific (see Table 2-24)	The value of the AIB attribute that was read.

2.2.4.4.2.2 When Generated

This primitive is generated by the APSME and issued to its next higher layer in response to an APSME-GET.request primitive. This primitive returns a status of SUCCESS, indicating that the request to read an AIB attribute was successful, or an error code of UNSUPPORTED_ATTRIBUTE. The reasons for these status values are fully described in Table 2-29.

2.2.4.4.2.3 Effect on Receipt

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

2.2.4.4.3 APSME-SET.request

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

2.2.4.4.3.1 Semantics of the Service Primitive

The semantics of this primitive are as follows:

```
APSME-SET.request            {
                             AIBAttribute,
                             AIBAttributeLength,
                             AIBAttributeValue
                             }
```


Table 2-12 specifies the parameters for this primitive.

Table 2-12. APSME-SET.request Parameters

Name	Type	Valid Range	Description
AIBAttribute	Integer	See Table 2-24	The identifier of the AIB attribute to be written.
AIBAttributeLength	Integer	0x0000 - 0xffff	The length, in octets, of the attribute value being set.
AIBAttributeValue	Various	Attribute-specific (see Table 2-24).	The value of the AIB attribute that SHOULD be written.

2.2.4.4.3.2 When Generated

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

2.2.4.4.3.3 Effect on Receipt

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

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

2.2.4.4.4 APSME-SET.confirm

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

2.2.4.4.4.1 Semantics of the Service Primitive

The semantics of this primitive are as follows:

```
APSME-SET.confirm          {
                           Status,
                           AIBAttribute
                           }
```

Table 2-13 specifies the parameters for this primitive.

Table 2-13. APSME-SET.confirm Parameters

Name	Type	Valid Range	Description
Status	Enumeration	SUCCESS, INVALID_PARAMETER, or UNSUPPORTED_ATTRIBUTE	The result of the request to write the AIB Attribute.
AIBAttribute	Integer	See Table 2-24.	The identifier of the AIB attribute that was written.

2.2.4.4.4.2 When Generated

This primitive is generated by the APSME and issued to its next higher layer in response to an APSME-SET.request primitive. This primitive returns a status of either SUCCESS, indicating that the requested value was written to the indicated AIB attribute, or an error code of INVALID_PARAMETER or UNSUPPORTED_ATTRIBUTE. The reasons for these status values are completely described in Table 2-29.

2.2.4.4.4.3 Effect on Receipt

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

2.2.4.5 Group Management

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

2.2.4.5.1 APSME-ADD-GROUP.request

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

2.2.4.5.1.1 Semantics of the Service Primitive

The semantics of this primitive are as follows:

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

Table 2-14 specifies the parameters for this primitive.

Table 2-14. APSME-ADD-GROUP.request Parameters

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

2.2.4.5.1.2 When Generated

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

2.2.4.5.1.3 Effect on Receipt

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

After checking the parameters as described above, the APSME will check the group table to see if an entry already exists containing the values given by the GroupAddress and Endpoint parameters. If such an entry already exists in the table then the APSME will issue the APSME-ADD-GROUP.confirm primitive to the next higher layer with a status value of SUCCESS. If there is no such entry and there is space in the table for another entry then the APSME will add a new entry to the group table with the values given by the GroupAddress and Endpoint parameters. The APSME then issues the APSME-ADD-GROUP.confirm primitive to the next higher layer with a status value of SUCCESS. If no entry for the given GroupAddress and Endpoint is present but there is no room in the group table for another entry, then the APSME will issue the APSME-ADD-GROUP.confirm primitive to the next higher layer with a status value of TABLE_FULL.

2.2.4.5.2 APSME-ADD-GROUP.confirm

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

2.2.4.5.2.1 Semantics of the Service Primitive

The semantics of the service primitive are as follows:

```
APSME-ADD-GROUP.confirm        {
                               Status,
                               GroupAddress,
                               Endpoint
                               }
```

Table 2-15 specifies the parameters for this primitive.

Table 2-15. APSME-ADD-GROUP.confirm Parameters

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

2.2.4.5.2.2 When Generated

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

2.2.4.5.2.3 Effect on Receipt

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

2.2.4.5.3 APSME-REMOVE-GROUP.request

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

2.2.4.5.3.1 Semantics of the Service Primitive

The semantics of the service primitive are as follows:

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

Table 2-16 specifies the parameters for this primitive.

Table 2-16. APSME-REMOVE-GROUP.request Parameters

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

2.2.4.5.3.2 When Generated

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

2.2.4.5.3.3 Effect on Receipt

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

After checking the parameters as described above, the APSME will check the group table to see if an entry exists containing the values given by the GroupAddress and Endpoint parameters. If such an entry already exists in the table, then that entry will be removed. Then, the APSME issues the APSME-REMOVE-GROUP.confirm primitive to the next higher layer with a status value of SUCCESS. If there is no such entry, the APSME will issue the APSME-REMOVE-GROUP.confirm primitive to the next higher layer with a status value of INVALID_GROUP.

2.2.4.5.4 APSME-REMOVE-GROUP.confirm

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

2.2.4.5.4.1 Semantics of the Service Primitive

The semantics of the service primitive are as follows:

```
APSME-REMOVE-GROUP.confirm        {
                                  Status,
                                  GroupAddress,
                                  Endpoint
                                  }
```

Table 2-17 specifies the parameters for this primitive.

Table 2-17. APSME-REMOVE-GROUP.confirm Parameters

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

2.2.4.5.4.2 When Generated

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

2.2.4.5.4.3 Effect on Receipt

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

2.2.4.5.5 APSME-REMOVE-ALL-GROUPS.request

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

2.2.4.5.5.1 Semantics of the Service Primitive

The semantics of the service primitive are as follows:

```
APSME-REMOVE-ALL-GROUPS.request        {
                                       Endpoint
                                       }
```

Table 2-18 specifies the parameters for this primitive.

Table 2-18. APSME-REMOVE-ALL-GROUPS.request Parameters

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

2.2.4.5.5.2 When Generated

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

2.2.4.5.5.3 Effect on Receipt

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

After checking the Endpoint parameter as described above, the APSME will remove all entries related to this endpoint from the group table. Then, the APSME issues the APSME-REMOVE-ALL-GROUPS.confirm primitive to the next higher layer with a status value of SUCCESS.

2.2.4.5.6 APSME-REMOVE-ALL-GROUPS.confirm

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

2.2.4.5.6.1 Semantics of the Service Primitive

The semantics of the service primitive are as follows:

```
APSME-REMOVE-ALL-GROUPS.confirm          {
                                         Status,
                                         Endpoint
                                         }
```

Table 2-19 specifies the parameters for this primitive.

Table 2-19. APSME-REMOVE-ALL-GROUPS.confirm Parameters

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

2.2.4.5.6.2 When Generated

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

2.2.4.5.6.3 Effect on Receipt

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

2.2.5 Frame Formats

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

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

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

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

2.2.5.1 General APDU Frame Format

The APS frame format is composed of an APS header and an APS payload. The fields of the APS header appear in a fixed order, however, the addressing fields may not be included in all frames. The general APS frame SHALL be formatted as illustrated in Figure 2-3.

<table>
<tr><th>Octets: 1</th><th>0/1</th><th>0/2</th><th>0/2</th><th>0/2</th><th>0/1</th><th>1</th><th>0/ Variable</th><th>Variable</th></tr>
<tr><td rowspan="2">Frame control</td><td>Destination endpoint</td><td>Group address</td><td>Cluster identifier</td><td>Profile identifier</td><td>Source endpoint</td><td rowspan="2">APS counter</td><td rowspan="2">Extended header</td><td rowspan="2">Frame payload</td></tr>
<tr><td colspan="5">Addressing fields</td></tr>
<tr><td colspan="8">APS header</td><td>APS payload</td></tr>
</table>

Figure 2-3. General APS Frame Format

2.2.5.1.1 Frame Control Field

The frame control field is 8 bits in length and contains information defining the frame type, addressing fields, and other control flags. The frame control field SHALL be formatted as illustrated in Figure 2-4.

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

Figure 2-4. Format of the Frame Control Field

2.2.5.1.1.1 Frame Type Sub-Field

The frame type sub-field is two bits in length and SHALL be set to one of the non-reserved values listed in Table 2-20.

Table 2-20. Values of the Frame Type Sub-Field

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

2.2.5.1.1.2 Delivery Mode Sub-Field

The delivery mode sub-field is two bits in length and SHALL be set to one of the non-reserved values from Table 2-21.

Table 2-21. Values of the Delivery Mode Sub-Field

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

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

If the value is 0b10, the message is a broadcast. In this case, the message will go to all devices defined for the selected broadcast address in use as defined in section 3.6.6. The destination endpoint field SHALL be set to a value between 0x01-0xfe (for broadcasts to specific endpoints) or to 0xff (for broadcasts to all active endpoints).

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

2.2.5.1.1.3 ACK Format Field

This bit indicates if the destination endpoint, cluster identifier, profile identifier and source endpoint fields SHALL be present in the acknowledgement frame. This is set to 0 for data frame acknowledgement and 1 for APS command frame acknowledgement.

2.2.5.1.1.4 Security Sub-Field

The Security Services Provider (see Chapter 4) manages the security sub-field.

2.2.5.1.1.5 Acknowledgement Request Sub-Field

The acknowledgement request sub-field is one bit in length and specifies whether the current transmission requires an acknowledgement frame to be sent to the originator on receipt of the frame. If this sub-field is set to 1, the recipient

SHALL construct and send an acknowledgement frame back to the originator after determining that the frame is valid. If this sub-field is set to 0, the recipient SHALL NOT send an acknowledgement frame back to the originator.

This sub-field SHALL be set to 0 for all frames that are broadcast or multicast.

2.2.5.1.1.6 Extended Header Present

The extended header present sub-field is one bit in length and specifies whether the extended header SHALL be included in the frame. If this sub-field is set to 1, then the extended header SHALL be included in the frame. Otherwise, it SHALL NOT be included in the frame.

2.2.5.1.2 Destination Endpoint Field

The destination endpoint field is 8-bits in length and specifies the endpoint of the final recipient of the frame. This frame SHALL be included in the frame only if the delivery mode subfield is set to 0b00 (normal unicast delivery), or 0b10 (broadcast delivery). In the case of broadcast delivery, the frame SHALL be delivered to the destination endpoint specified within the range 0x01-0xfe or to all active endpoints if specified as 0xff.

A destination endpoint value of 0x00 addresses the frame to the Zigbee device object (ZDO), resident in each device. A destination endpoint value of 0x01-0xfe addresses the frame to an application operating on that endpoint. A destination endpoint value of 0xff addresses the frame to all active endpoints except endpoint 0x00.

2.2.5.1.3 Group Address Field

The group address field is 16 bits in length and will only be present if the delivery mode sub-field of the frame control has a value of 0b11. In this case, the destination endpoint SHALL NOT be present. If the APS header of a frame contains a group address field, the frame will be delivered to all endpoints for which the group table in the device contains an association between that endpoint and the group identified by the contents of the group address field.

2.2.5.1.4 Cluster Identifier Field

The cluster identifier field is 16 bits in length and specifies the identifier of the cluster to which the frame relates and which SHALL be made available for filtering and interpretation of messages at each device that takes delivery of the frame. This field SHALL be present only for data or acknowledgement frames.

2.2.5.1.5 Profile Identifier Field

The profile identifier is two octets in length and specifies the Zigbee profile identifier for which the frame is intended and SHALL be used during the filtering of messages at each device that takes delivery of the frame. This field SHALL be present only for data or acknowledgement frames.

2.2.5.1.6 Source Endpoint Field

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

2.2.5.1.7 APS Counter

This field is eight bits in length and is used as described in section 2.2.8.4.2 to prevent the reception of duplicate frames. This value SHALL be incremented by one for each new transmission.

2.2.5.1.8 Extended Header Sub-Frame

The extended header sub-frame contains further sub-fields and SHALL be formatted as illustrated in Figure 2-5.

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

Figure 2-5. Format of the Extended Header Sub-Frame

2.2.5.1.8.1 Extended Frame Control Field

The extended frame control field is eight bits in length and contains information defining the use of fragmentation. The extended frame control field SHALL be formatted as illustrated in Figure 2-6.

Bits: 0-1	2-7
Fragmentation	Reserved

Figure 2-6. Format of the Extended Frame Control Field

The fragmentation sub-field is two bits in length and SHALL be set to one of the non-reserved values listed in Table 2-22.

Table 2-22. Values of the Fragmentation Sub-Field

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

2.2.5.1.8.2 Block Number

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

2.2.5.1.8.3 ACK Bitfield

The ACK bitfield field is one octet in length and is used in an APS acknowledgement as described in section 2.2.8.4.3 to indicate which blocks of a fragmented ASDU have been successfully received. This field is only present if the frame type sub-field indicates an acknowledgement and the fragmentation sub-field indicates a fragmented transmission.

2.2.5.1.9 Frame Payload Field

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

2.2.5.2 Format of Individual Frame Types

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

2.2.5.2.1 Data Frame Format

The data frame SHALL be formatted as illustrated in Figure 2-7.

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

Figure 2-7. Data Frame Format

The order of the fields of the data frame SHALL conform to the order of the general APS frame as illustrated in Figure 2-8.

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

In the frame control field, the frame type sub-field SHALL contain the value that indicates a data frame, as shown in Table 2-20. All other sub-fields SHALL be set appropriately according to the intended use of the data frame.

2.2.5.2.1.1 Data Payload Field

For an outgoing data frame, the data payload field SHALL contain part or all of the sequence of octets that the next higher layer has requested the APS data service to transmit. For an incoming data frame, the data payload field SHALL contain all or part of the sequence of octets that has been received by the APS data service and that is to be delivered to the next higher layer.

2.2.5.2.2 APS Command Frame Format

The APS command frame SHALL be formatted as illustrated in Figure 2-8.

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

Figure 2-8. APS Command Frame Format

The order of the fields of the APS command frame SHALL conform to the order of the general APS frame as illustrated in .

2.2.5.2.2.1 APS Command Frame APS Header Fields

The APS header field for an APS command frame SHALL contain the frame control and APS counter fields. In this version of the specification, the APS command frame SHALL NOT be fragmented and the extended header field SHALL NOT be present.

In the frame control field, the frame type sub-field SHALL contain the value that indicates an APS command frame, as shown in Table 2-20. The APS Command Payload SHALL be set appropriately according to the intended use of the APS command frame.

2.2.5.2.2.2 APS Command Identifier Field

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

2.2.5.2.2.3 APS Command Payload Field

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

2.2.5.2.3 Acknowledgement Frame Format

The acknowledgement frame SHALL be formatted as illustrated in Figure 2-9.

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

Figure 2-9. Acknowledgement Frame Format

The order of the fields of the acknowledgement frame SHALL conform to the order of the general APS frame as illustrated in Figure 2-3.

2.2.5.2.3.1 Acknowledgement Frame APS Header Fields

If the ACK format field is not set in the frame control field, the destination endpoint, cluster identifier, profile identifier and source endpoint SHALL be present. This is not set for data frame acknowledgement. The extended header field SHALL be included in a data frame according to the value of the extended header present sub-field of the frame control field.

In the frame control field, the frame type sub-field SHALL contain the value that indicates an acknowledgement frame, as shown in Table 2-20. The extended header present sub-field SHALL contain the same value as in the frame to which this frame is an acknowledgement. All other sub-fields shall be set appropriately according to the intended use of the acknowledgement frame.

If the ACK format field is set in the frame control field, the frame is an APS command frame acknowledgement and the destination endpoint, cluster identifier, profile identifier and source endpoint fields SHALL NOT be included. Alternatively, if an APS data frame is being acknowledged, the source endpoint field SHALL reflect the value in the destination endpoint field of the frame that is being acknowledged. Similarly, the destination endpoint field SHALL reflect the value in the source endpoint field of the frame that is being acknowledged. And the Cluster identifier and Profile identifier fields SHALL contain the same values as in the frame to which this frame is an acknowledgement.

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

Where the extended header is present, the fragmentation sub-field of the extended frame control field SHALL contain the same value as in the frame to which this frame is an acknowledgement. If fragmentation is in use for this frame, then the block number and ACK bitfield fields SHALL be present. Where present, the block number field SHALL contain block number to which this frame is an acknowledgement. If fragmentation is in use, the acknowledgement frames SHALL be issued according to section 2.2.8.4.5.4 and not for each received frame unless the transmission window size is set to request acknowledgement of each frame.

2.2.6 Command Frames

This specification defines no command frames. Refer to section 4.4.11 for a thorough description of the APS command frames and primitives related to security.

2.2.7 Constants and PIB Attributes

2.2.7.1 APS Constants

The constants that define the characteristics of the APS sub-layer are presented in Table 2-23.

Table 2-23. APS Sub-Layer Constants

Constant	Description	Value
apscMaxDescriptorSize	The maximum number of octets contained in a non-complex descriptor.	64
apscMaxFrameRetries	The maximum number of retries allowed after a transmission failure.	3
apscAckWaitDuration	The maximum number of seconds to wait for an acknowledgement to a transmitted frame.	1.6 seconds 0.05 * (2* *nwkcMaxDepth*) + (security encrypt/decrypt delay) where the (security encrypt/decrypt delay) = 0.1 (assume 0.05 per encrypt or decrypt cycle)
apscMinDuplicateRejectionTableSize	The minimum required size of the APS duplicate rejection table.	1

Constant	Description	Value
apscMinHeaderOverhead	The minimum number of octets added by the APS sub-layer to an ASDU.	12
apsParentAnnounceBaseTimer	The base amount of delay, in seconds, before each broadcast parent announce is sent.	10 seconds
apsParentAnnounceJitterMax	The max amount of jitter that is added to the apsParentAnnounceBaseTimer before each broadcast parent announce is sent.	10 seconds.
apscJoinerTLVsUnfragmentedMaxSize	The max amount of TLV payload that an parent router can pass from the Joiner to the Trust Center. This value only applies to the TLV payload of the APS Command: Update Device.	79
apscMaxWindowSize	The stack-wide window size supported by all endpoints used by the application.	1
apscInterframeDelay	Fragmentation parameter—the standard delay, in milliseconds, between sending two blocks of a fragmented transmission (see section 2.2.8.4.5). This parameter is global to the stack across all endpoints.	0 (Not used for Window Size 1)

2.2.7.2 APS Information Base

The APS information base comprises the attributes required to manage the APS layer of a device. The attributes of the AIB are listed in Table 2-24. The security-related AIB attributes are described in section 4.4.12.

Table 2-24. APS IB Attributes

Attribute	Identifier	Type	Range	Description	Default
apsBindingTable	0xc1	Set	Variable	The current set of binding table entries in the device (see section 2.2.8.2.1).	Null set
apsDesignated-Coordinator	0xc2	Boolean	TRUE or FALSE	TRUE if the device SHOULD become the Zigbee Coordinator on startup, FALSE if otherwise.	FALSE

Attribute	**Identifier**	**Type**	Range	**Description**	**Default**
apsChannelMask-List	0xc3	List of IEEE Std 802.15.4 channel masks	Any legal list of masks for the PHY	The list of masks of allowable channels for this device to use for network operations.	All channels
apsUseExtended-PANID	0xc4	64-bit extended address	0x0000000000000000 to 0xfffffffffffffffe	The 64-bit address of a network to form or to join.	0x0000000000000000
apsGroupTable	0x0c5	Set	Variable	The current set of group table entries (see Table 2-25).	Null set
Reserved	0xc6				
apsUseInsecure-Join	0xc8	Boolean	TRUE or FALSE	A flag controlling the use of insecure join at startup.	FALSE
apsInter-frameDelay	0xc9	Integer	0x00 to 0xff (MAY be restricted by application profile)	Fragmentation parameter—the standard delay, in milliseconds, between sending two blocks of a fragmented transmission (see section 2.2.8.4.5).	Set by application profile
apsLastChannel Energy	0xca	Integer	0x00 - 0xff	The energy measurement for the channel energy scan performed on the previous channel just before a channel change (in accordance with [B1]).	Null set
apsLastChannel FailureRate	0xcb	Integer	0-100 (decimal)	The latest percentage of transmission network transmission failures for the previous channel just before a channel change (in percentage of failed transmissions to the total number of transmissions attempted).	Null set

Attribute	**Identifier**	**Type**	Range	**Description**	**Default**
apsChannelTimer	0xcc	Integer	1-24 (decimal)	A countdown timer (in hours) indicating the time to the next permitted frequency agility channel change. A value of NULL indicates the channel has not been changed previously.	Null set
apsParent AnnounceTimer	0xce	Integer	0 to apsParentAnnounceBaseTimer + apsParentAnnounceJitterMax	The value of the current countdown timer before the next Parent_annce is sent.	0
apsZdoRestrictedMode	0xcf	Boolean	TRUE or FALSE	Indicates whether or not the ZDO is in restricted mode and thus will not accept changes to its configuration. TRUE indicates certain ZDO commands will not be accepted unless sent by Trust Center with APS encryption. FALSE indicates that other nodes on the network may change the configuration of the device (e.g. bindings).	FALSE
apsStatTable	0xd0	See Table 2-26.	-	A table of statistics from the APS, NWK, MAC, and Security layers.	0
apsFragmentationCacheTable	0xd1	See Table 2-27.	-	The set of stored fragmentation parameters for other devices in the network.	
apsFragmentationCacheSize	0xd2	Integer	0 – 65,535	The number of entries the apsFragmentationCacheTable supports	1

Attribute	Identifier	Type	Range	Description	Default
apsMaxSizeASDU	0xd3	Integer	0 – 65,535	The Maximum Incoming Transfer Unit supported by the stack across all endpoints. This indicates the maximum reassembled message size at the application layer after fragmentation has been applied on the message at the lower layers. A device supporting fragmentation would set this field to be larger than the normal payload size of the underlying NWK and MAC layer.	128+
apsZdoResponseTimeout	0xd4	Integer	0 – 255	The amount of time to wait in seconds before the stack will timeout ZDO requests and issue a APSDE-DATA.confirm with the result.	3
apsApplicationFragmentationSupport	0xd5	Boolean	TRUE or FALSE	This bit is set by the higher application layer to indicate whether fragmentation is supported. This is separate from the stack's ability to support fragmentation of ZDO messages.	FALSE

Table 2-25. Group Table Entry Format

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

2.2.7.2.1 Statistics Table

The AIB SHALL maintain a statistics table with the items detailed in Table 2-26. These statistics MAY require the AIB to probe the lower layers in order to obtain the data.

All statistics are kept until they are reset via the next higher layer. Statistics do not need to be kept in non-volatile storage.

2309 **Table 2-26. Statistics Table (apsStatTable)**

Statistic Name	Type	Range	Description
apsTxUnicastSuccess	Integer	0 – 65,535	The total number of successful APS unicast messages. APS messages without APS Acks SHALL be considered successfully sent when the MAC ACK from the next hop is received. APS messages with APS Acks SHALL be considered successful when the APS ACK is received for the message.
apsTxUnicastRetry	Integer	0 – 65,535	The total number of APS retries that have been recorded.
apsTxUnicastFailures	Integer	0 – 65,535	The total number of APS unicast messages that are considered failed. APS unicast messages without APS Acks shall be considered failed when the MAC ACK from the next hop is not received after all MAC and NWK layer retries. APS unicast messages with APS Acks shall be considered failed when the APS ACK is not received for the message after all APS layer retries.
nwkFrameCounterFailures	Integer	0 – 65,535	The total number of received messages dropped due to a frame counter failure at the network security layer.
apsFrameCounterFailures	Integer	0 – 65,535	The total number of received messages dropped due to a frame counter failure at the APS security layer.
apsUnauthorizedKey	Integer	0 – 65,535	The total number of received messages dropped at the APS security layer because the key is not authorized.
nwkDecryptFailures	Integer	0 – 65,535	This is the total number of decryption failures at the NWK security layer.
apsDecryptFailures	Integer	0 – 65,535	This is the total number of decryption failures at the APS security layer.
bufferAllocationFailures	Integer	0 – 65,535	This is the total number of failures by the stack due to a lack of memory buffers.
phyToMacQueueFailures	Integer	0 – 65,535	This is the total number of failures by the stack to transfer a message from the PHY layer to the MAC layer.

Statistic Name	Type	Range	Description
packetValidationFailures	Integer	0 – 65,535	This is the total number of packets dropped due to invalid formatting.
macTxUcastSuccess	Integer	0 – 65,535	The total number of successful mac unicast transmissions.
macTxUcastRetry	Integer	0 – 65,535	The total number of retries at the MAC layer for unicast messages. This includes retries within an MCPS transaction and not just the final MCPS result.
macTxUcastFail	Integer	0 – 65,535	The total number of MAC unicast failures. This includes failures within an MCPS transaction and not just the final MCPS result.

2.2.7.2.2 Fragmentation Cache

The *apsFragmentationCacheTable* is a table with a number of entries equal to apsFragmentationCacheSize. It has the following elements in each entry in Table 2-27.

Table 2-27. apsFragmentationCacheTable Entry Elements

Name	Type	Description
DestinationEUI64	EUI64	The long address of the device associated with this entry.
MaxIncomingTxSize	0 – 65,535	The maximum incoming transmission size of the APDU for the associated device. This determines the size of a reassembled message the device can receive.
Supported	Boolean	This indicates support for fragmentation with the standard R23 fragmentation parameters.

Implementations MAY choose to combine the data in the *apsFragmentationCacheTable* with the *apsDeviceKeyPairEntries* table.

2.2.8 Functional Description

2.2.8.1 Persistent Data

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

- *apsBindingTable* (if supported on the device)
- *apsDesignatedCoordinator* (if supported on the device)
- *apsChannelMaskList*
- *apsUseExtendedPANID*

- *apsUseInsecureJoin*
- *apsGroupTable* (if supported on the device)
- Node Descriptor, Power Descriptor plus the Simple Descriptor(s) for each active endpoint on the device
- Network manager address

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

2.2.8.2 Binding

The APS MAY maintain a binding table, which allows Zigbee devices to establish a designated destination for frames from a given source endpoint and with a given cluster ID. Each designated destination SHALL represent either a specific endpoint on a specific device, or a group address.

2.2.8.2.1 Binding Table Implementation

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

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

Where:

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

2.2.8.2.2 Binding

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

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

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

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

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

The procedure for a successful binding is illustrated in the MSC shown in Figure 2-10.

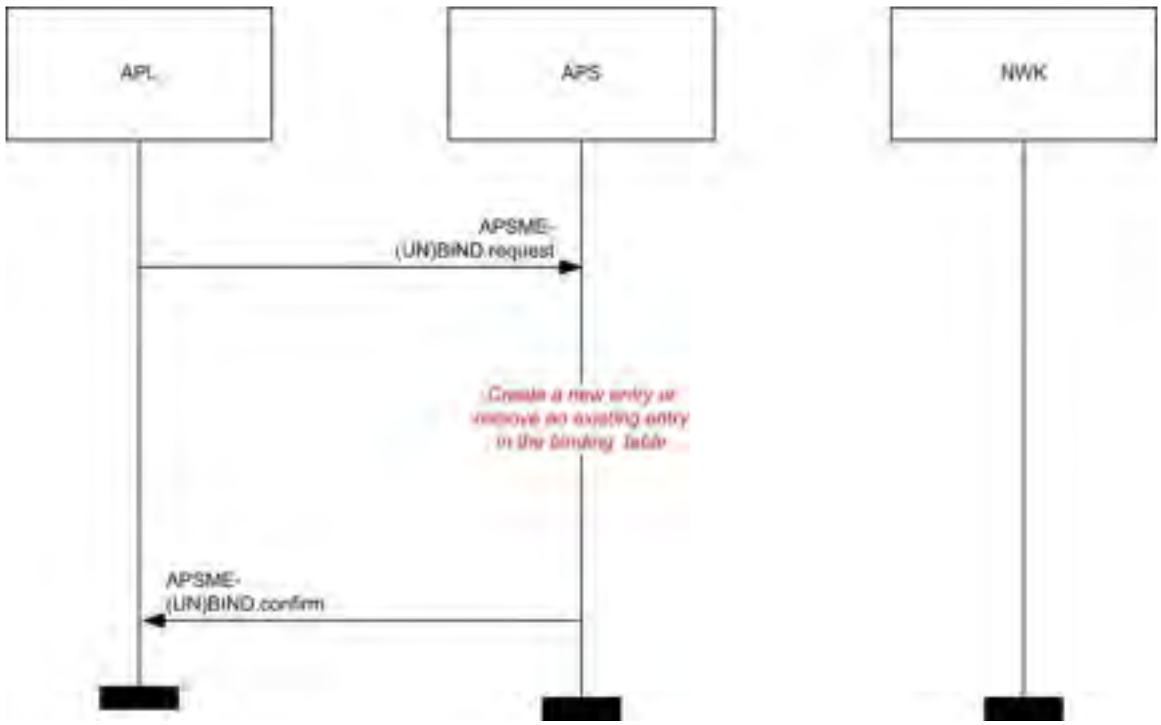


Figure 2-10. Binding on a Device Supporting a Binding Table

2.2.8.3 Group Addressing

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

2.2.8.3.1 The Group Table

For purposes of this discussion, the group table SHALL be viewed as a set of associations between groups and end-points as follows:

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

Where:

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

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

2.2.8.4 Transmission, Reception, and Acknowledgement

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

2.2.8.4.1 Transmission

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

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

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

For all devices where the transmission is due to a binding table entry stored on the source device, the APSDE of the source device SHALL determine whether the binding table entry contains a unicast destination device address or a destination group address. In the case where a binding table entry contains a unicast destination device address and this destination device address is that of the source device itself, the APSDE SHALL issue an APSDE-DATA.indication primitive to the next higher layer and SHALL NOT transmit a frame. Otherwise, the APSDE SHALL transmit the frame to the 16-bit NWK address corresponding to the destination address indicated by the binding table entry, and the delivery mode sub-field of the frame control field SHALL be set to 0b00. In the case where the binding table entry contains a destination group address, the delivery mode sub-field of the frame control field SHALL have a value of 0b11, the destination group address SHALL be placed in the APS header, and the destination endpoint SHALL be omitted. The frame SHALL then be broadcast using the NLDE-DATA.request primitive and employing a broadcast address of 0xfffd.

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

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

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

2.2.8.4.2 Reception and Rejection

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

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

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

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

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

2.2.8.4.3 Use of Acknowledgements

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

2.2.8.4.3.1 No Acknowledgement

A frame that is received by its intended recipient with its acknowledgement request (AR) sub-field set to 0 SHALL NOT be acknowledged. The originating device SHALL assume that the transmission of the frame was successful. Figure 2-11 shows the scenario for transmitting a single frame of data from an originator to a recipient without requiring an acknowledgement. In this case, the originator transmits the data frame with the AR sub-field equal to 0.

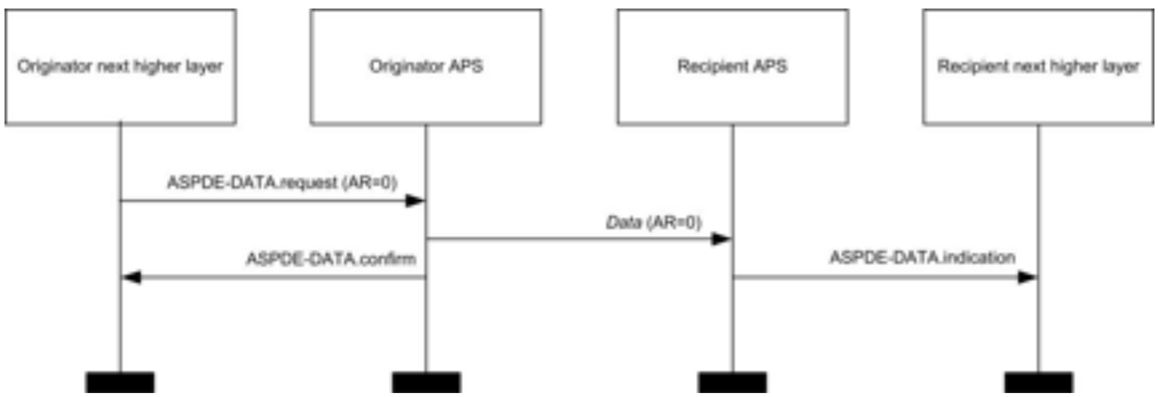


Figure 2-11. Successful Data Transmission Without an Acknowledgement

2.2.8.4.3.2 Acknowledgement

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

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

Figure 2-12 shows the scenario for transmitting a single frame of data from an originator to a recipient with an acknowledgement. In this case, the originator indicates to the recipient that it requires an acknowledgement by transmitting the data frame with the AR sub-field set to 1.

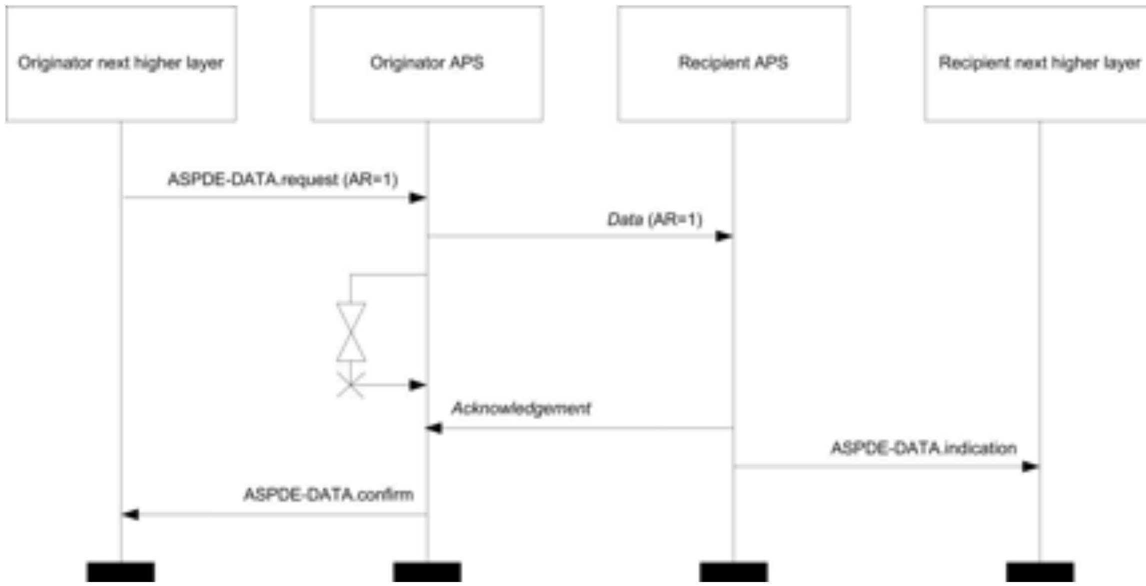


Figure 2-12. Successful Data Transmission with an Acknowledgement

2.2.8.4.4 Retransmissions

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

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

If an acknowledgement frame is received within *apscAckWaitDuration* seconds, containing the same cluster identifier and APS counter as the original frame and has a source endpoint equal to the destination endpoint to which the original frame was transmitted, the transmission SHALL be considered successful and no further action SHALL be taken by the device. If an acknowledgement is not received within *apscAckWaitDuration* seconds, or an acknowledgement is received within *apscAckWaitDuration* seconds but contains an unexpected cluster identifier or APS counter or has a source endpoint that is not equal to the destination endpoint to which the original frame was transmitted, the device SHALL conclude that the single transmission attempt has failed.

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

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

2.2.8.4.5 Fragmented Transmissions

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

A "transmission window" is used to arrange an orderly transaction. The window size is set by the stack profile, and MAY be set as high as eight blocks. The protocol below arranges that all blocks in a transmission window SHALL be received and acknowledged before the window can move on. An acknowledgement is sent when all blocks in the transmission window have been successfully received or, according to the protocol below, to request retransmission of one or more unreceived blocks.

Transactions not using APS acknowledgements MAY not be fragmented. Multicast and broadcast transmissions are not permitted to use fragmentation.

The use of a window size of 1 is the only window size guaranteed to interoperate. All stacks implementing Revision 23 of this specification SHALL support window size of 1. All window sizes greater than 1 are a manufacturer specific feature.

2.2.8.4.5.1 Fragmentation Discovery & Caching

All nodes starting with Revision 23 of this specification SHALL support fragmented transmissions using a standard set of fragmentation parameters as noted in section 2.2.8.4.5.2. However, devices supporting earlier specification re-visions were per-mitted to not support fragmentation with these same parameters. Support is determined by querying the Node Descriptor of the target device to determine whether the target device can receive fragmented transmissions and what is the maximum incoming transfer size for their ASDU.

The stack is required to automatically determine support for the target device when the message submitted to the APSDE-DATA.request primitive requires fragmentation. The results of the queries can be stored in the apsFragmentationCacheTable, and it is up to the implementation to decide how to manage that cache.

The APSDE-DATA.confirm SHALL report back failed results to discover the cache size, or when the discovered cache size is too small for the requested message to be transmitted.

Due to the importance of the Trust Center in the network it is required to have an apsFragmentationCacheTableSize equal to the number of apsDeviceKeyPairEntries in the network. In other words, the Trust Center has enough cache to keep track of all devices in the network.

2.2.8.4.5.2 Fragmentation Parameters

All Revision 23 and later devices SHALL support and advertise the fragmentation parameters detailed in the AIB. They are repeated Table 2-28 for clarity.

Table 2-28. Fragmentation Parameters

Parameter	Value
apscWindowSize	1
apscInterframeDelay	*Not Applicable for Window Size 1*
apscMaxASDU	128 bytes MINIMUM[1]

[1] The stack MAY implement a size larger than this, but MUST implement this value as a minimum.

2.2.8.4.5.3 Transmission

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

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

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

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

On receipt of an acknowledgement frame with matching values in the APS counter, block number, and addressing fields, outgoing blocks are acknowledged as described in the section below. If at least one previously unacknowledged block is acknowledged, then the timer SHALL be stopped and the retryCounter parameter reset. If all blocks in the current transmission window have been acknowledged, then the transmission window SHALL be increased by *apscMaxWindowSize*. If all blocks have now been transmitted and acknowledged, then the transaction is complete, and an APSDE-DATA.confirm primitive SHALL be returned to the NHLE with a status value of SUCCESS. Otherwise, the block with the lowest unacknowledged block number SHALL be passed to the NWK data service.

2.2.8.4.5.4 Reception and Rejection, and Acknowledgements

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

If an incoming fragmented transaction is already in progress but the addressing and APS counter fields do not match those of the received frame, then the received frame MAY optionally be rejected or handled independently as a further transaction.

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

If a transaction is initiated with APS counter and source address field values matching a previously received transaction, then the new transaction MAY be rejected as a duplicate.

Upon receipt of the first received block (not necessarily block 0) in the first window, or when an acknowledgement is generated, the receiver SHALL set a timer for *apscAckWaitDuration*.

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

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

Additionally an APS acknowledgement SHALL be generated for the window if any one of the following circumstances occurs: (1) the last block in the entire fragmented transmission is received, (2) the last block in the window is received, (3) a block is received and all subsequent blocks in the window have been previously received and acknowledged. If a block is received with its block number value outside of the current window, then an acknowledgement SHALL NOT be generated.

Once all blocks in the transaction have been received, the APS SHALL issue an APSDE-DATA.indication primitive containing the reassembled message, and the transaction SHALL be deemed to be complete. A period of persistence of *apscAckWaitDuration* seconds is encouraged in order to facilitate any retransmission of the final acknowledgement.

Where generated, the acknowledgement is formatted according to the acknowledgement frame format specified in section 2.2.5.2.3 and SHALL include the extended APS header.[1] The APS counter field SHALL reflect the value in the corresponding field of the frame(s) being acknowledged. The block number field SHALL contain the value of the lowest block number in the current receive window, using the value 0 as the value of the first block.

The first bit of the ACK bitfield SHALL be set to 1 if the first fragment in the current receive window has been correctly received, and 0 otherwise. Subsequent bits SHALL be set similarly, with values corresponding to subsequent fragments in the current receive window. If *apsMaxWindowSize* is less than 8, then the remaining bits SHALL be set to 1.

The process is illustrated in the following diagrams. In Figure 2-13, the transmission is successful immediately. (These examples assume that *apscMaxWindowSize* takes the value 3).

[1] CCB 1571

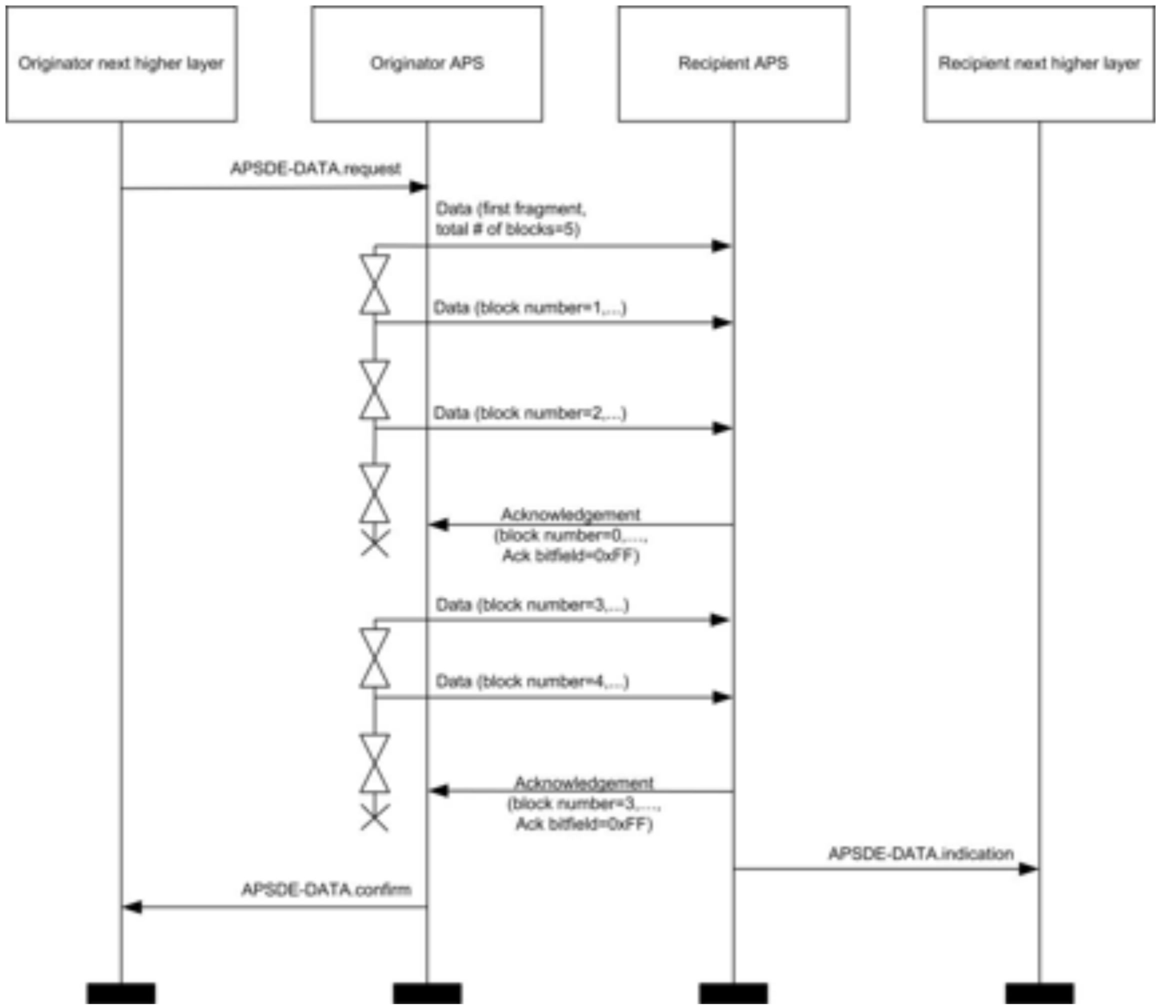


Figure 2-13. Successful Data Transmission with Fragmentation

In Figure 2-14, a single frame is lost during transit across the network, and is retransmitted.

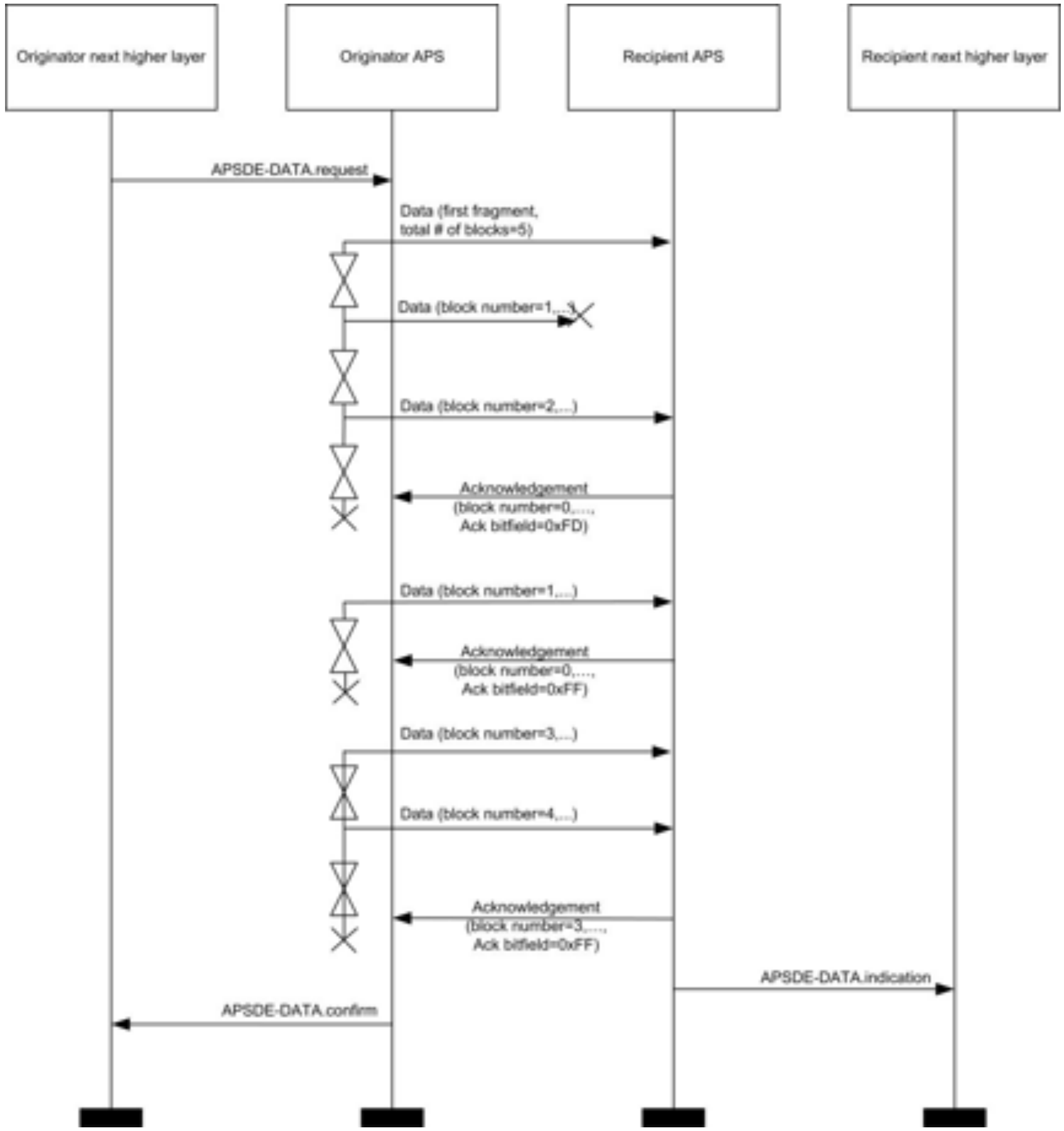


Figure 2-14. Fragmented Data Transmission with a Single Retransmission

In Figure 2-15, multiple frames are lost in the network, including a frame which has the highest block number in the window. Slightly more traffic is required in this case, but the source backs off and gives the network a chance to recover, and the ASDU is delivered successfully.

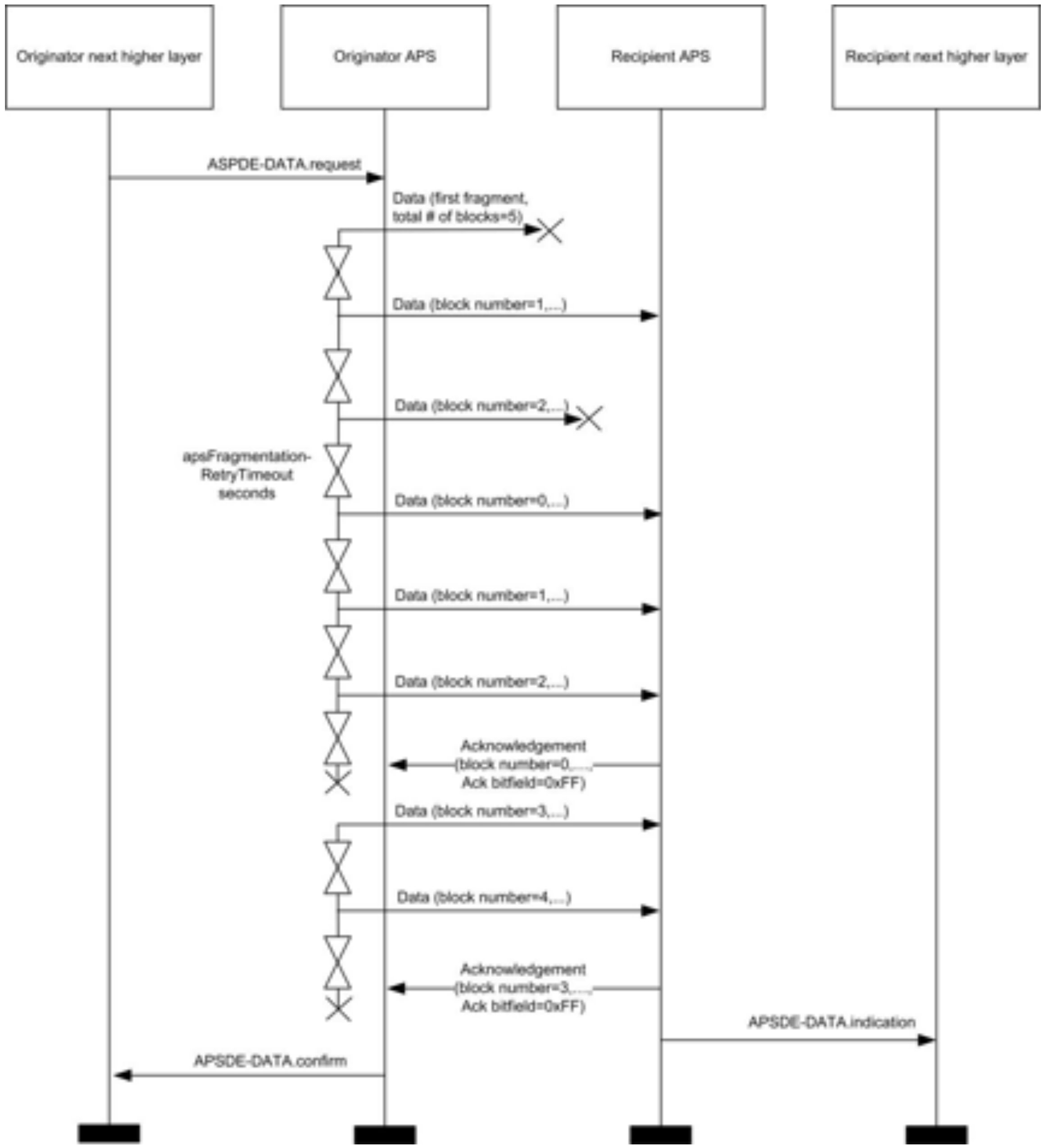


Figure 2-15. Fragmented Data Transmission with Multiple Retransmissions

2.2.9 APS Sub-Layer Status Values

Application support (APS) sub-layer confirm primitives often include a parameter that reports on the status of the request to which the confirmation applies. Values for APS sub-layer Status parameters appear in Table 2-29.

2579 **Table 2-29. APS Sub-layer Status Values**

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

Name	Value	Description
TABLE_FULL	0xae	An APSME-BIND.request or APSME.ADD-GROUP.request issued when the binding or group tables, respectively, were full.
UNSECURED	0xaf	An ASDU was received without any security.
UNSUPPORTED_ATTRIBUTE	0xb0	An APSME-GET.request or APSME-SET.request has been issued with an unknown attribute identifier.
PEER_CANNOT_FRAGMENT	0xb1	The target device cannot receive fragmented transmissions and the ASDU requires fragmentation.
UNKNOWN_FRAGMENT_SUP-PORT	0xb2	The target device did not respond to a discovery request of its fragmentation parameters.

2.3 The Zigbee Application Framework

2.3.1 Creating a Zigbee Profile

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

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

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

2.3.1.1 Getting a Profile Identifier from the Connectivity Standards Alliance

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

- Device descriptions
- Cluster identifiers

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

2.3.1.2 Defining Device Descriptions and Clusters

The profile identifier is the main enumeration feature within the Zigbee protocol. Each unique profile identifier defines an associated enumeration of device descriptions and cluster identifiers. For example, for profile identifier “1”, there exists a pool of device descriptions described by a 16-bit value (meaning there are 65,536 possible device descriptions within each profile) and a pool of cluster identifiers described by a 16-bit value (meaning there are 65,536 possible cluster identifiers within each profile). Each cluster identifier also supports a pool of attributes described by a 16-bit value. As such, each profile identifier has up to 65,536 cluster identifiers and each of those cluster identifiers contains up to 65,536 attributes. It is the responsibility of the profile developer to define and allocate device descriptions, cluster identifiers, and attributes within their allocated profile identifier. Note that the definition of device descriptions, cluster identifiers, and attribute identifiers SHALL be undertaken with care to ensure efficient creation of simple descriptors and simplified processing when exchanging messages.

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

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

2.3.1.3 Deploying the Profile on Endpoints

A single Zigbee device MAY contain support for many profiles, provide for subsets of various cluster identifiers defined within those profiles, and MAY support multiple device descriptions. This capability is defined using a hierarchy of addressing within the device as follows:

- **Device:** The entire device is supported by one or more radios and has just one unique IEEE and NWK address.
- **Endpoints:** This is an 8-bit field that describes different applications that are supported by a single radio. Endpoint 0x00 is used to address the device profile, which each Zigbee device SHALL employ, endpoint 0xff is used to address all active endpoints (the broadcast endpoint). Consequently, a single physical Zigbee device can support up to 254 applications on endpoints 0x01-0xfe. Note that endpoints 0xf1-0xfe can only be used for Connectivity Standards Alliance approved applications.

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

2.3.1.4 Enabling Service Discovery

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

One important point is that service discovery is made on the basis of profile identifier, input cluster identifier list, and output cluster identifier list (device description is notably missing). The device description is simply a convention for specifying mandatory and optional cluster identifier support within devices of that type for the indicated profile. Additionally, it is EXPECTED that the device description enumeration would be employed within PDAs or other assisted binding devices to provide external descriptions of device capabilities.

2.3.1.5 Mixing Standard and Proprietary Profiles

As an example, a Zigbee device could be developed to Zigbee public profile identifier "XX." If a manufacturer wanted to deploy a Zigbee device supporting public profile "XX" and also provide manufacturer specific extensions, these extensions could be added to the manufacturer's implementation of public profile "XX" if manufacturer extensions are supported within the definition of profile "XX." Alternatively, if manufacturer extensions are not supported or the type of desired manufacturer extensions aren't supported in profile "XX," the manufacturer MAY deploy the extensions in a separate manufacturer-specific profile identifier advertised on a separate endpoint within the same physical device. In either case, devices that support the profile identifier "XX" but not containing the manufacturer extensions, would only advertise support for the base features of public profile identifier "XX" and could not respond to or create messages using the manufacturer extensions.

2.3.1.6 Enabling Backward Compatibility

In the previous example, a device is created using Zigbee public profile identifier "XX." If the Connectivity Standards Alliance were to update this public profile at a later time to add new features, the revisions could either be incorporated directly into public profile identifier "XX" if such extensions are supported via the definition of the profile, or could be introduced into a new public profile with a new profile identifier (say "XY"). Assuming extensibility is not supported in public profile "XX," devices manufactured with just profile identifier "XX" could still be compatible with newer devices manufactured later by having the newer devices advertise support for both profile identifier "XX" and profile identifier "XY." In this manner, the newer device MAY communicate with older devices using profile identifier "XX"; however, it MAY also communicate with newer devices using profile identifier "XY" from within the same application. The service discovery feature within Zigbee enables devices on the network to determine the level of support.

It is the goal of the Connectivity Standards Alliance to provide extensibility, both for manufacturer extensions to public profiles as well as future enhancements to public profiles. That goal includes maintaining those extensions and enhancements within the same profile identifier whenever possible. This section illustrates that the profile definition features within Zigbee permit deployment of manufacturer extensions and feature enhancements, whether the goal of profile extensibility is achievable or not. The subject of profile extensibility, both for manufacturer extensions and feature enhancements, is beyond the scope of this document and addressed in other Alliance documents.

2.3.2 Zigbee Descriptors

Zigbee devices describe themselves using descriptor data structures. The actual data contained in these descriptors is defined in the individual device descriptions. There are three descriptors: node, node power, and simple shown in Table 2-30.

Table 2-30. Zigbee Descriptors

Descriptor Name	Status	Description
Node	M	Type and capabilities of the node.
Node power	M	Node power characteristics.
Simple	M	Device descriptions contained in node.
Complex	Deprecated	This descriptor has been deprecated.
User	Deprecated	This descriptor has been deprecated.

2.3.2.1 Transmission of Descriptors

The node, node power, and simple descriptors SHALL be transmitted in the order that they appear in their respective tables, i.e., the field at the top of the table is transmitted first and the field at the bottom of the table is transmitted last. Each individual field SHALL follow the transmission order specified in section 1.2.3.

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

2.3.2.2 Discovery via Descriptors

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

The node, node power, complex, and user descriptors apply to the complete node. The simple descriptor SHALL be specified for each endpoint defined in the node. If a node contains multiple subunits, these will be on separate end-points and the specific descriptors for these endpoints are read by including the relevant endpoint number in the Zigbee device profile primitive.

2.3.2.3 Node Descriptor

The node descriptor contains information about the capabilities of the Zigbee node and is mandatory for each node. There SHALL be only one node descriptor in a node. All reserved and deprecated bits SHALL be set to zero.

The fields of the node descriptor are shown in Table 2-31 in their order of transmission.

Table 2-31. Fields of the Node Descriptor

Field Name	Length (Bits)
Logical type	3
Deprecated	1
Deprecated	1
Fragmentation Supported (R23)	1
Reserved	2
APS flags	3
Frequency band	5
MAC capability flags	8
Manufacturer code	16
Maximum buffer size	8
Maximum incoming transfer size	16
Server mask	16

Field Name	Length (Bits)
Maximum outgoing transfer size	16
Descriptor capability field	8

2.3.2.3.1 Logical Type Field

The logical type field of the node descriptor is three bits in length and specifies the device type of the Zigbee node. The logical type field SHALL be set to one of the non-reserved values listed in Table 2-32.

Table 2-32. Values of the Logical Type Field

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

2.3.2.3.2 Complex Descriptor Field – Deprecated

2.3.2.3.3 User Descriptor Field – Deprecated

2.3.2.3.4 Fragmentation Supported Field

This field was added in Revision 23 of the specification. When examining a Node Descriptor received over the air from another device, this field SHALL only be examined when the Stack Compliance Revision within the Server Mask field is set to 23 or higher.

This field indicates whether the device supports fragmentation at the APS layer. The maximum size of reassembled message that can be received is reflected in the Maximum incoming transfer size field of the Node Descriptor.

If this field is set to 1 and the Stack Compliance Revision is 23 or greater, then the device has support for APS layer fragmentation. If this field is set to 0 and the Stack Compliance Revision is 23 or greater, then the device does not have support for APS layer fragmentation. If the Stack Compliance Revision is less than 23, the support of fragmentation must be determined via other means.

2.3.2.3.5 APS Flags Field

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

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

2.3.2.3.6 Frequency Band Field

The frequency band field of the node descriptor is five bits in length and specifies the frequency bands that are supported by the underlying IEEE Std 802.15.4 radio(s) utilized by the node. For each frequency band supported by any physically present underlying IEEE Std 802.15.4 radio, the corresponding bit of the frequency band field, as listed in Table 2-33, SHALL be set to 1. All other bits SHALL be set to 0.

Table 2-33. Values of the Frequency Band Field

Frequency Band Field Bit Number	Supported Frequency Band
0	868 – 868.6 MHz
1	Reserved
2	902 – 928 MHz
3	2400 – 2483.5 MHz
4	GB Smart Energy sub-GHz bands: (863-876MHz and 915-921MHz)

2.3.2.3.7 MAC Capability Flags Field

The MAC capability flags field is eight bits in length and specifies the node capabilities, as required by the IEEE Std 802.15.4-2020 MAC sub-layer [B1]. The MAC capability flags field SHALL be formatted as illustrated in Figure 2-16.

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

Figure 2-16. Format of the MAC Capability Flags Field

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

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

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

The receiver on when idle sub-field is one bit in length and SHALL be set to 1 if the device does not disable its receiver to conserve power during idle periods. Otherwise, the receiver on when idle sub-field SHALL be set to 0 (see also Table 2-36).

The security capability sub-field is one bit in length and SHALL be set to 1 if the device is capable of sending and receiving frames secured using the security suite specified in section 4.2.2. Otherwise, the security capability sub-field SHALL be set to 0.

The allocate address sub-field is one bit in length and SHALL be set to1 on transmission and ignored on reception

2.3.2.3.8 Manufacturer Code Field

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

2.3.2.3.9 Maximum Buffer Size Field

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

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

2.3.2.3.10 Maximum Incoming Transfer Size Field

This indicates the device's apsMaxSizeASDU AIB value.

The maximum transfer size field of the node descriptor is sixteen bits in length, with a valid range of 0x0000-0x7fff. This field specifies the maximum size, in octets, of the application sub-layer data unit (ASDU) that can be transferred to this node in one single message transfer. This value can exceed the value of the node maximum buffer size field (see section 2.3.2.3.9) through the use of fragmentation.

2.3.2.3.11 Server Mask Field

The server mask field of the node descriptor is sixteen bits in length, with bit settings signifying the system server capabilities of this node. It is used to facilitate discovery of particular system servers by other nodes on the system. The bit settings are defined in Table 2-34.

Table 2-34. Server Mask Bit Assignments

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

2.3.2.3.11.1 Stack Compliance Revision

These bits indicate the Revision of the Zigbee Pro Core specification that the running stack is implemented to. Prior to Revision 21 of the specification these bits were reserved and thus set to 0. A stack that is compliant to Revision 23 would set these bits to 23 (0010111b). A stack SHALL indicate the Revision of the specification it is compliant to by setting these bits.

2.3.2.3.12 Maximum Outgoing Transfer Size Field

The maximum transfer size field of the node descriptor is sixteen bits in length, with a valid range of 0x0000-0x7fff. This field specifies the maximum size, in octets, of the application sub-layer data unit (ASDU) that can be transferred from this node in one single message transfer. This value can exceed the value of the node maximum buffer size field (see section 2.3.2.3.9) through the use of fragmentation.

2.3.2.3.13 Descriptor Capability Field – Deprecated

2.3.2.4 Node Power Descriptor

The node power descriptor gives a dynamic indication of the power status of the node and is mandatory for each node. There SHALL be only one node power descriptor in a node. This data has been superseded by the Zigbee Cluster Library Power Configuration Cluster. This Node Descriptor SHOULD NOT be used.

The fields of the node power descriptor are shown in Table 2-35 in the order of their transmission.

Table 2-35. Fields of the Node Power Descriptor

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

2.3.2.4.1 Current Power Mode Field

The current power mode field of the node power descriptor is four bits in length and specifies the current sleep/power-saving mode of the node. The current power mode field SHALL be set to one of the non-reserved values listed in Table 2-36.

Table 2-36. Values of the Current Power Mode Field

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

2.3.2.4.2 Available Power Sources Field

The available power sources field of the node power descriptor is four bits in length and specifies the power sources available on this node. For each power source supported on this node, the corresponding bit of the available power sources field, as listed in Table 2-37, SHALL be set to 1. All other bits SHALL be set to 0.

Table 2-37. Values of the Available Power Sources Field

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

2.3.2.4.3 Current Power Source Field

The current power source field of the node power descriptor is four bits in length and specifies the current power source being utilized by the node. For the current power source selected, the corresponding bit of the current power source field, as listed in Table 2-38, SHALL be set to 1. All other bits SHALL be set to 0.

Table 2-38. Values of the Current Power Sources Field

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

2.3.2.4.4 Current Power Source Level Field

The current power source level field of the node power descriptor is four bits in length and specifies the level of charge of the power source. The current power source level field SHALL be set to one of the non-reserved values listed in Table 2-39.

Table 2-39. Values of the Current Power Source Level Field

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

2.3.2.5 Simple Descriptor

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

The fields of the simple descriptor are shown in Table 2-40 in their order of transmission. As this descriptor needs to be transmitted over air, the overall length of the simple descriptor SHALL be less than or equal to *apscMaxDescriptorSize*.

Table 2-40. Fields of the Simple Descriptor

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

2.3.2.5.1 Endpoint Field

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

2.3.2.5.2 Application Profile Identifier Field

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

2.3.2.5.3 Application Device Identifier Field

The application device identifier field of the simple descriptor is sixteen bits in length and specifies the device description supported on this endpoint. Device description identifiers SHALL be obtained from the Connectivity Standards Alliance.

2.3.2.5.4 Application Device Version Field

The application device version field of the simple descriptor is four bits in length and specifies the version of the device description supported on this endpoint. The application device version field SHALL be set to one of the non-reserved values listed in Table 2-41.

Table 2-41. Values of the Application Device Version Field

Application Device Version Value $b_3b_2b_1b_0$	Description
0000 – 1111	Specific values to be set by the application profile described by the application profile identifier in this descriptor. Default SHALL be 0000 unless otherwise defined by the application profile.

2.3.2.5.5 Application Input Cluster Count Field

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

2.3.2.5.6 Application Input Cluster List

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

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

2.3.2.5.7 Application Output Cluster Count Field

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

2.3.2.5.8 Application Output Cluster List

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

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

2.3.2.6 Complex Descriptor – Deprecated

2.3.2.7 User Descriptor – Deprecated

2.3.3 Functional Description

2.3.3.1 Reception and Rejection

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

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

If the application framework receives a frame for an inactive endpoint, the frame SHALL be discarded. Otherwise, if the profile identifier passes the Profile Id Endpoint Matching Rules (see section 2.3.3.3). The application framework SHALL pass the payload of the received frame to the application implemented on the specified endpoint.

When a message originates from an endpoint implemented using the public Profile ID, the profile ID in the simple descriptor SHALL be used. If the recipient of the message is able to process the message, it SHALL respond with the same profile ID that it received in the request.

It is permissible for the originator of the message to send its messages with a wild card profile ID. The recipient of the message containing a request using a wild card profile ID SHALL respond with the profile ID in its simple descriptor if it is able to process the message.

2.3.3.2 ZDO Messages

ZDO message transmission and reception rules are as described in the relevant ZDO server chapters.

2.3.3.3 Application Endpoints Using Manufacturer Specific Profiles

Application endpoints using Manufacturer Specific Profiles SHALL NOT use the wild card profile ID for transmission. They SHALL transmit with the profile ID of the simple descriptor and SHALL respond with the profile ID of the simple descriptor.

2.3.4 PAN ID Conflicts

2.3.4.1 Detecting and Reporting via the ZDO

PAN ID conflicts are an unusual event in a network. Legitimate PAN ID conflicts may occur when IEEE Std 802.15.4 networks grow over time and inadvertently collide. However, it is also possible that malicious devices may try to trigger PAN ID conflicts in order to disrupt the network’s operations.

In previous versions of the specification, an immediate detect and respond approach was codified whereby conflicts would trigger a network wide change. This approach has been revisited and the specification now discourages this due to the fact that PAN ID conflicts can be falsely reported.

An application can still query devices about detected conflicts and gather statistics over time in order to inform whether a persistent PAN ID conflict is occurring. It is highly recommended that application connectivity problems be a factor in the decision to change PAN IDs, and not simply the presence of an apparent conflict.

2.3.4.2 Unsolicited PAN ID Conflict Reports from the Network Layer

Starting with Revision 23 of this specification, devices SHALL no longer report PAN ID conflicts immediately as they are detected. Instead, devices will count these conflicts and store that data in the NIB. The NIB data may be retrieved via a Security_Get_Configuration_req by requesting the PAN ID Conflict Report Global TLV.

Older devices will still send unsolicited PAN ID conflict reports to the Network Manager. For a Revision 23 Network Manager receiving a report, this will be indicated to the local application via the NLME-NETWORK-STATUS.indication with a status of 0x14 (PAN ID Conflict Report). This data can be used to help determine whether a network will change its PAN ID, but it SHOULD NOT be the sole reason for that change.

2.3.4.3 Querying Devices for Conflicts

A Network Manager application may periodically query router devices for this information. Each Security_Get_Configuration_req will trigger a reset of the NIB value. This will allow the Network Manager application to gather statistics over a period of time to determine if there is a persistent PAN ID problem.

In addition, the Mgmt_Beacon_Survey_req can be used to trigger an active scan on the target device and the PAN ID Conflict Report Global TLV will be returned as well, without resetting the nwkPanIdConflictCount NIB value.

The exact rules by which the application determines to change PAN IDs is outside the scope of this specification. The expectation of the applications running on the network and the impact on sleepy devices SHOULD be considered before any change to PAN ID is made.

2.4 The Zigbee Device Profile

2.4.1 Scope

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

2.4.2 Device Profile Overview

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

-
-
- Network Management Overview

2.4.2.1 Device and Service Discovery Overview

Device and Service Discovery are distributed operations where individual devices respond to discovery requests.

The following capabilities exist for device and service discovery:

Device Discovery: Provides the ability for a device to determine the identity of other devices on the PAN. Device Discovery is supported for both the 64-bit IEEE address and the 16-bit Network address.

Device Discovery messages can be used in one of two ways:

- **Broadcast addressed:** All devices on the network SHALL respond according to the Logical Device Type and the matching criteria. Zigbee End Devices SHALL respond with just their address. Zigbee Coordinators and Zigbee Routers with associated devices SHALL respond with their address as the first entry followed by the addresses of their associated devices depending on the type of request. The responding devices SHALL employ APS acknowledged service on the unicast responses.
- **Unicast addressed:** Only the specified device responds. A Zigbee End Device SHALL respond only with its address. A Zigbee Coordinator or Router SHALL reply with its own address and the address of each associated child device. Inclusion of the associated child devices allows the requestor to determine the network topology underlying the specified device.

Service Discovery: Provides the ability for a device to determine services offered by other devices on the PAN.

Service Discovery messages can be used in one of two ways:

- **Broadcast addressed:** Due to the volume of information that could be returned, only the individual device SHALL respond with the matching criteria established in the request. The responding devices SHALL also employ APS acknowledged service on the unicast responses.
- **Unicast addressed:** Only the specified device SHALL respond.

Service Discovery is supported with the following query types:

- **Active Endpoint:** This command permits an enquiring device to determine the active endpoints. An active endpoint is one with an application supporting a single profile, described by a Simple Descriptor. The command SHALL be unicast addressed.
- **Match Simple Descriptor:** This command permits enquiring devices to supply a Profile ID (and, optionally, lists of input and/or output Cluster IDs) and ask for a return of the identity of an endpoint on the destination device which matches the supplied criteria. This command MAY be broadcast to all devices for which

macRxOnWhenIdle = TRUE, or unicast addressed. For broadcast addressed requests, the responding device SHALL employ APS acknowledged service on the unicast responses.

- **Simple Descriptor:** This command permits an enquiring device to return the Simple Descriptor for the supplied endpoint. This command SHALL be unicast addressed.
- **Node Descriptor:** This command permits an enquiring device to return the Node Descriptor from the specified device. This command SHALL be unicast addressed.
- **Power Descriptor:** This command permits an enquiring device to return the Power Descriptor from the specified device. This command SHALL be unicast addressed.
- **Complex Descriptor:** This optional command permits an enquiring device to return the Complex Descriptor from the specified device. This command SHALL be unicast addressed.
- **User Descriptor:** This optional command permits an enquiring device to return the User Descriptor from the specified device. This command SHALL be unicast addressed.

2.4.2.2 End Device Bind Overview – Deprecated

2.4.2.3 Bind and Unbind Overview

The following capabilities exist for directly configuring binding table entries:

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

2.4.2.4 Binding Table Management Overview – Deprecated

2.4.2.5 Network Management Overview

The following capabilities exist for network management:

Provides the ability to retrieve management information from the devices including:

- Network discovery results
- Link quality to neighbor nodes
- Routing table contents
- Binding table contents
- Energy detection scan results
- Provides the ability to set management information controls including:
- Network leave
- Network direct join
- Permit joining
- Network update and fault notification

2.4.2.6 Device Descriptions for the Device Profile

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

2.4.2.7 Configuration and Roles

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

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

The Device Profile describes devices in one of two configurations:

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

2.4.2.8 Transmission of ZDP Commands

All ZDP commands shall be transmitted via the APS data service and SHALL be formatted according to the ZDP frame structure, as illustrated in Figure 2-17.

Octets: 1	**Variable**
Transaction sequence number	Transaction data

Figure 2-17. Format of the ZDP Frame

2.4.2.8.1 Transaction Sequence Number Field

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

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

The transaction sequence number field can be used by a controlling device, which MAY have issued multiple commands, so that it can match the incoming responses to the relevant command.

2.4.2.8.2 Transaction Data Field

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

2.4.2.8.3 Fragmentation of ZDO Messages

Zigbee Devices based on Revision 22 or earlier do not handle fragmentation of ZDO commands. Except for the commands listed in Table 2-42. ZDO commands SHALL NOT be fragmented in transmission.

Table 2-42. ZDO Commands Permitted to be Fragmented

Cluster ID	Name
0x0040	Security_Start_Key_Negotiation_req
0x0041	Security_Retrieve_Authentication_Token_req
0x0043	Security_Set_Configuration_req
0x8040	Security_Start_Key_Negotiation_rsp

Cluster ID	Name
0x8041	Security_Retrieve_Authentication_Token_rsp
0x8043	Security_Set_Configuration_rsp

Devices supporting the commands in Table 2-42 SHALL be able to handle fragmentation and reassembly of these commands. The required parameters for fragmentation are specified in section 2.2.8.4.5.2. Additionally, the Fragmentation Parameters Global TLV can be used to advertise a device's capabilities. This TLV is mandatory to be included in various messages as described in the description of those ZDO messages.

A sending device SHALL determine the receiving device's fragmentation capabilities prior to sending it a fragmentation transmission. For devices already on the network this can be done by querying the Node Descriptor using the Node_Desc_req. For devices not on the network yet, the Trust Center includes the Fragmentation Parameters Global TLV in the set of TLVs advertised in the Beacons of the Network. This is updated in all routers via the Mgmt_Permit_Joining_req.

A device sending a ZDO Response SHALL assume the device that sent the request can support fragmentation. The device sending the response determines the fragmented transmission size based on its capabilities, the requestor's capabilities, and the default minimum.

The Responder SHALL determine the maximum incoming transfer size of the Requester in the following way.

1. If the Requester provided a Fragmentation Parameters Global TLV in the request, the Maximum Incoming Transfer Size from the TLV SHALL be used. If it is not provided in the request, the default Maximum Incoming transfer size of 128 bytes SHALL be used.
2. Compare the value determined in step 1 to the device's local Maximum Outgoing Transfer Size. Take the smaller of the two values.

If the response is larger than the requesting device can handle, then a ZDO response with a status of FRAME_TOO_LARGE is generated.

For example, Device A sends a ZDO_Security_Get_Configuration_req and indicates via the Fragmentation Parameters Global TLV it supports up to 200 bytes for its Maximum Incoming Transfer Size. Device B prepares a ZDO Security_Get_Configuration_rsp and examines its own local Maximum Outgoing Transfer Size, which is 300. It uses the smaller value of 200 indicated by Device A when fragmenting the transmission. If the response would exceed Device A's smaller value it would instead generate a ZDO Security_Get_Configuration_rsp with a status of FRAME_TOO_LARGE.

2.4.2.8.4 APS Acknowledgements

All unicast ZDO Command request and responses SHALL set the Acknowledgement request sub-field of the APS Frame control. This will enable ZDO messages to overcome transient routing or buffering failures in the network. This SHALL be done by submitting a APSDE-DATA.request with the TxOptions including 0x04 in the value. When the ZDO message allows fragmentation the options SHALL also include 0x08, fragmentation permitted.

2.4.3 Client Services

The Device Profile Client Services support the transport of device and service discovery requests, bind requests, unbind requests, and network management requests from client to server. Additionally, Client Services support receipt of responses to these requests from the server.

Restricted Mode (apsZdoRestrictedMode) is a mode where a device will conditionally accept specific ZDO commands, depending on the restricted criteria, source address, and encryption policy of the incoming command. If a command is accepted, it is subject to normal command processing. The acceptance criteria is explain further below:

1. If the command is marked as "Yes" in the *Restricted Command* column, do the following:
    a. If *apsZdoRestrictedMode* in the AIB is set to FALSE, the command is not restricted.
        i. Go to Step 2.
    b. If the sender is the Trust Center AND has APS encryption, the command is not restricted.

i. Go to Step 2.

c. Otherwise, the command SHALL NOT be processed. The receiver SHALL do the following:

i. If the command was broadcast, no error is generated.

1. No more processing is done.

ii. If the command was unicast, generate an error message. Create the corresponding ZDO Response frame with a status of NOT_AUTHORIZED.

1. No more processing is done.

2. Continue processing the command normally.

2.4.3.1 Device and Service Discovery Client Services

Table 2-43 lists the commands supported by Device Profile, Device, and Service Discovery Client Services. Each of these commands will be discussed in the following sections.

Table 2-43. Device and Service Discovery Client Services Commands

Device and Service Discovery Client Services	Cluster ID	Client Transmission	Server Processing	Restricted Command
NWK_addr_req	0x0000	O	M	No
IEEE_addr_req	0x0001	O	M	No
Node_Desc_req	0x0002	M	M	No
Power_Desc_req	0x0003	O	M	No
Simple_Desc_req	0x0004	O	M	No
Active_EP_req	0x0005	O	M	No
Match_Desc_req	0x0006	O	M	No
Complex_Desc_req	0x0010	Deprecated	Deprecated	-
User_Desc_req	0x0011	Deprecated	Deprecated	-
Discovery_Cache_req	0x0012	Deprecated	Deprecated	-
Device_annce	0x0013	O	M	No
Parent_annce	0x001F	M	M	No
User_Desc_set	0x0014	Deprecated	Deprecated	-
System_Server_Discovery_req	0x0015	O	M	No
Discovery_store_req	0x0016	Deprecated	Deprecated	-
Node_Desc_store_req	0x0017	Deprecated	Deprecated	-

Device and Service Discovery Client Services	Cluster ID	Client Transmission	Server Processing	Restricted Command
Power_Desc_store_req	0x0018	Deprecated	Deprecated	-
Active_EP_store_req	0x0019	Deprecated	Deprecated	-
Simple_Desc_store_req	0x001a	Deprecated	Deprecated	-
Remove_node_cache_req	0x001b	Deprecated	Deprecated	-
Find_node_cache_req	0x001c	Deprecated	Deprecated	-
Extended_Simple_Desc_req	0x001d	Deprecated	Deprecated	-
Extended_Active_EP_req	0x001e	Deprecated	Deprecated	-

2.4.3.1.1 NWK_addr_req

The NWK_addr_req command (ClusterID=0x0000) SHALL be formatted as illustrated in Figure 2-18.

Octets: 8	1	1
IEEEAddress	RequestType	StartIndex

Figure 2-18. Format of the NWK_addr_req Command Frame

Table 2-44 specifies the fields of the NWK_addr_req Command Frame.

Table 2-44. Fields of the NWK_addr_req Command Frame

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

2.4.3.1.1.1 When Generated

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

2.4.3.1.1.2 Effect on Receipt

Upon receipt, a Remote Device SHALL compare the IEEEAddr to its *nwkIeeeAddress* in the NIB or any IEEE address held in its *nwkNeighborTable* where the Device Type field of the entry is 0x02 (End Device). If there is no match and the request was unicast, a NWK_addr_rsp command SHALL be generated and sent back to the local device with the Status field set to DEVICE_NOT_FOUND, the IEEEAddrRemoteDev field set to the IEEE address of the request; the NWKAddrRemoteDev field set to 0xFFFF indicating that there is no known short address; and the NumAssocDev, StartIndex, and NWKAddrAssocDevList fields SHALL NOT be included in the frame. If there is no match and the command was received as a broadcast, the request SHALL be discarded and no response generated. Note that router parent *and* the macRxOnWhenIdle=TRUE end device SHALL *both* respond to the NWK Address request when the request is sent to the macRxOnWhenIdle=TRUE broadcast address.

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

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

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

2.4.3.1.2 IEEE_addr_req

The IEEE_addr_req command (ClusterID=0x0001) SHALL be formatted as illustrated in Figure 2-19.

Octets: 2	**1**	**1**
NWKAddrOfInterest	RequestType	StartIndex

Figure 2-19. Format of the IEEE_addr_req_Command Frame

Table 2-45 specifies the fields of the IEEE_addr_req command frame.

Table 2-45. Fields of the IEEE_addr_req Command

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

2.4.3.1.2.1 When Generated

The IEEE_addr_req is generated from a Local Device wishing to inquire as to the 64-bit IEEE address of the Remote Device based on their known 16-bit address. The destination addressing on this command SHALL be unicast. or broadcast to all devices for which macRxOnWhenIdle = TRUE.

2.4.3.1.2.2 Effect on Receipt

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

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

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

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

2.4.3.1.3 Node_Desc_req

The Node_Desc_req_command (ClusterID=0x0002) SHALL be formatted as illustrated in Figure 2-20.

Octets: 2	Octets: Variable
NWKAddrOfInterest	TLVs

Figure 2-20. Format of the Node_Desc_req Command Frame

Table 2-46 specifies the fields for the Node_Desc_req command frame.

Table 2-46. Fields of the Node_Desc_req Command Frame

Name	Type	Valid Range	Description
NWKAddrOfInterest	Device Address	16-bit NWK address	NWK address for the request
TLVs	Concatenation of TLVs	Varies	The Fragmentation Parameters Global TLV SHALL always be included. If the Node_Desc_req is sent to the Trust Center from a device wishing to update its Trust Center link-key, the Supported Key Negotiation Methods Global TLV (ID 65) SHALL be included.

2.4.3.1.3.1 When Generated

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

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

The Fragmentation Parameters Global TLV SHALL be present to indicate the sending device's fragmentation capabilities. This allows the receiving device to cache the information if it needs to.

If the Node_Desc_req is sent to the Trust Center from a device wishing to update its Trust Center link-key, the Supported Key Negotiation Methods Global TLV (ID 65) SHALL be included.

2.4.3.1.3.2 Effect on Receipt

Upon receipt of this command, the recipient device SHALL process the command and generate a Node_Desc_rsp command in response, according to the description in section 2.4.4.2.3.

2.4.3.1.4 Power_Desc_req

The Power_Desc_req command (ClusterID=0x0003) SHALL be formatted as illustrated in Figure 2-21.

Octets: 2
NWKAddrOfInterest

Figure 2-21. Format of the Power_Desc_req Command Frame

Table 2-47 specifies the fields of the Power_Desc_req command frame.

Table 2-47. Fields of the Power_Desc_req Command Frame

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

2.4.3.1.4.1 When Generated

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

The local device SHALL generate the Power_Desc_req command using the format illustrated in Table 2-47. The NWKAddrOfInterest field SHALL contain the network address of the remote device for which the power descriptor is required.

2.4.3.1.4.2 Effect on Receipt

Upon receipt of this command, the recipient device SHALL process the command and generate a Power_Desc_rsp command in response according to the description in section 2.4.4.2.4.

2.4.3.1.5 Simple_Desc_req

The Simple_Desc_req command (ClusterID=0x0004) SHALL be formatted as illustrated in Figure 2-22.

Octets: 2	1
NWKAddrOfInterest	EndPoint

Figure 2-22. Format of the Simple_Desc_req Command Frame

Table 2-48 specifies the fields of the Simple_Desc_req command frame.

Table 2-48. Fields of the Simple_Desc_req Command

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

2.4.3.1.5.1 When Generated

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

The local device SHALL generate the Simple_Desc_req command using the format illustrated in Table 2-48. The NWKAddrOfInterest field SHALL contain the network address of the remote device for which the simple descriptor is required and the endpoint field SHALL contain the endpoint identifier from which to obtain the required simple descriptor.

2.4.3.1.5.2 Effect on Receipt

Upon receipt of this command, the recipient device SHALL process the command and generate a Simple_Desc_rsp command in response, according to the description in section 2.4.4.2.5.

2.4.3.1.6 Active_EP_req

The Active_EP_req command (ClusterID=0x0005) SHALL be formatted as illustrated in Figure 2-23.

Octets: 2
NWKAddrOfInterest

Figure 2-23. Format of the Active_EP_req Command Frame

Table 2-49 specifies the fields of the Active_EP_req command frame.

Table 2-49. Fields of the Active_EP_req Command

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

2.4.3.1.6.1 When Generated

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

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

2.4.3.1.6.2 Effect on Receipt

Upon receipt of this command, the recipient device SHALL process the command and generate an Active_EP_rsp command in response, according to the description in section 2.4.4.2.6.

2.4.3.1.7 Match_Desc_req

The Match_Desc_req command (ClusterID=0x0006) SHALL be formatted as illustrated in Figure 2-24.

Octets: 2	2	1	Variable	1	Variable
NWKAddrOfInter-est	ProfileID	NumInClusters	InClusterList	NumOutClusters	OutClusterList

Figure 2-24. Format of the Match_Desc_req Command Frame

Table 2-50 specifies the fields of the Match_Desc_req command frame.

Table 2-50. Fields of the Match_Desc_req Command

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

2.4.3.1.7.1 When Generated

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

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

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

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

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

2.4.3.1.7.2 Effect on Receipt

Upon receipt of this command, the recipient device SHALL process the command and generate a Match_Desc_rsp command in response, according to the description in section 2.4.4.2.7.

2.4.3.1.8 Complex_Desc_req – DEPRECATED

2.4.3.1.9 User_Desc_req – DEPRECATED

2.4.3.1.10 Discovery_Cache_req – DEPRECATED

2.4.3.1.11 Device_annce

The Device_annce command (ClusterID=0x0013) SHALL be formatted as illustrated in Figure 2-25.

Octets: 2	**8**	**1**
NWKAddr	IEEEAddr	Capability

Figure 2-25. Format of the Device_annce Command Frame

Table 2-51 specifies the fields of the Device_annce command frame.

Table 2-51. Fields of the Device_annce Command

Name	Type	Valid Range	Description
NWKAddr	Device Address	16-bit NWK address	NWK address for the Local Device
IEEEAddr	Device Address	64-bit IEEE address	IEEE address for the Local Device
Capability	Bitmap	See Figure 2-16.	Capability of the local device

2.4.3.1.11.1 When Generated

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

2.4.3.1.11.2 Effect on Receipt

Routers and Coordinators SHALL first determine whether there is an address conflict with any other device on the network. End Devices are not required to detect address conflicts.

Address conflicts SHALL be determined as follows:

1. Using the value of the IEEEAddr in the received ZDO message examine NIB tables for the nwkAddressMap and nwkNeighborTable for a matching IEEE Address.
2. If a match is found AND that match has a different node ID than the value for NWKAddr in the received ZDO message then an address conflict has conflict has occurred. Do the following:
    a. If the conflicted entry is the nwkNeighborTable of the NIB AND the entry has a Relationship of 0x06, neighbor is a lost child, this indicates a local end device child has NOT changed parents and needs a new address. Perform an NLME-SET.req as follows.
        i. Set the corresponding entry in the nwkNeighborTable to have a Relationship field of 0x07, neighbor is a child that needs new address.
    b. Follow the procedure in section 3.6.1.10.3 to resolve the address conflict.
        i. No further processing of the message SHALL be done.

When no conflict is detected, all device types SHALL continue processing the ZDO device announce as indicated below.

Upon receipt, the Remote Device SHALL use the IEEEAddr in the message to find a match with any other IEEE address held in the Remote Device. If a match is detected, the Remote Device SHALL update the nwkAddressMap attribute of the NIB with the updated NWKAddr corresponding to the IEEEAddr received.

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

2.4.3.1.12 Parent_annce

The Parent_annce command (ClusterID = 0x001F) SHALL be formatted as illustrated in Figure 2-26.

Octets: 1	**Variable**	**…**	**Variable**
NumberOfChildren	ChildInfo[0]	…	ChildInfo[n]

Figure 2-26. Format of the Parent Annce Message

Table 2-52 specifies the contents of the ChildInfo structure.

Table 2-52. Format of the ChildInfo Structure

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

2.4.3.1.12.1 When Generated

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

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

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

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

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

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

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

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

If the device shall send multiple Parent_annce messages but receives a keepalive from an end device before it has sent the Parent_Annce message, it SHALL NOT include that device in the message.

2.4.3.1.12.2 Effect on receipt

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

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

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

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

If the NumberOfChildren field value is 0, the local device SHALL discard the previously constructed Parent_Annce_rsp. No response message SHALL be sent.

If the NumberOfChildren field in the Parent_Annce_rsp is greater than 0, it SHALL unicast the message to the sender of the Parent_Annce message.

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

2.4.3.1.13 User_Desc_set – DEPRECATED

2.4.3.1.14 System_Server_Discovery_req

The System_Server_Discovery_req command (ClusterID=0x0015) SHALL be formatted as illustrated in Figure 2-27.

Octets: 2
ServerMask

Figure 2-27. Format of the System_Server_Discovery_req Command Frame

Table 2-53 specifies the fields of the System_Server_Discovery_req command frame.

Table 2-53. Fields of the System_Server_Discovery_req Command Frame

Name	Type	Valid Range	Description
ServerMask	Bitmap	16 bits	See Table 2-34 for bit assignments.

2.4.3.1.14.1 When Generated

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

2.4.3.1.14.2 Effect on Receipt

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

2.4.3.1.15 Discovery_store_req – DEPRECATED

2.4.3.1.16 Node_Desc_store_req – DEPRECATED

2.4.3.1.17 Power_Desc_store_req – DEPRECATED

2.4.3.1.18 Active_EP_store_req – DEPRECATED

2.4.3.1.19 Simple_Desc_store_req – DEPRECATED

2.4.3.1.20 Remove_node_cache_req – DEPRECATED

2.4.3.1.21 Find_node_cache_req – DEPRECATED

2.4.3.1.22 Extended_Simple_Desc_req – DEPRECATED

2.4.3.1.23 Extended_Active_EP_req – DEPRECATED

2.4.3.2 Bind, Unbind, and Bind Management Client Services Primitives

Table 2-54 lists the primitives supported by Device Profile: Bind and Unbind Client Services. Each of these commands will be discussed in the following sections.

Table 2-54. Bind, Unbind, and Bind Management Client Service Commands

Bind and Unbind Client Services	Cluster ID	Client Transmission	Server Processing	Restricted Mode Only
End_Device_Bind_req	0x0020	Deprecated	Deprecated	-
Bind_req	0x0021	O	O	Yes
Unbind_req	0x0022	O	O	Yes
Bind_Register_req	0x0023	Deprecated	Deprecated	-
Replace_Device_req	0x0024	Deprecated	Deprecated	-
Store_Bkup_Bind_Entry_req	0x0025	Deprecated	Deprecated	-
Remove_Bkup_Bind_Entry_req	0x0026	Deprecated	Deprecated	-
Backup_Bind_Table_req	0x0027	Deprecated	Deprecated	-
Recover_Bind_Table_req	0x0028	Deprecated	Deprecated	-
Backup_Source_Bind_req	0x0029	Deprecated	Deprecated	-
Recover_Source_Bind_req	0x002a	Deprecated	Deprecated	-
Clear_All_Bindings_req	0x002b	O	M / O *	Yes

*The Clear_All_Bindings is optional if no binding table is present. If a Binding Table is supported then the Clear_All_Bindings_req command server processing is mandatory.

2.4.3.2.1 End_Device_Bind_req – DEPRECATED

2.4.3.2.2 Bind_req

The Bind_req command (ClusterID=0x0021) SHALL be formatted as illustrated in Figure 2-28.

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

Figure 2-28. Format of the Bind_req Command Frame

Table 2-55 specifies the fields of the Bind_req command frame.

Table 2-55. Fields of the Bind_req Command

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

2.4.3.2.2.1 When Generated

The Bind_req is generated from a Local Device wishing to create a Binding Table entry for the source and destination addresses contained as parameters. The destination addressing on this command SHALL be unicast only, and the destination address SHALL be that of the SrcAddress itself. The Binding Manager is optionally supported on the source device (unless that device is also the Zigbee Coordinator) so that device SHALL issue a NOT_SUPPORTED status to the Bind_req if not supported.

2.4.3.2.2.2 Effect on Receipt

On receipt of a broadcast Bind request the stack SHALL drop the message and no further processing SHALL take place. Otherwise, upon receipt, a Remote Device SHALL create a Binding Table entry based on the parameters supplied in the Bind_req if the Binding Manager is supported. The Remote Device SHALL then respond with SUCCESS if the entry has been created by the Binding Manager; otherwise, the Remote Device SHALL respond with INSUFFICIENT_SPACE.

2.4.3.2.3 Unbind_req

The Unbind_req command (ClusterID=0x0022) SHALL be formatted as illustrated in Figure 2-29.

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

Figure 2-29. Format of the Unbind_req Command Frame

Table 2-56 specifies the fields of the Unbind_req command frame.

Table 2-56. Fields of the Unbind_req Command

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

2.4.3.2.3.1 When Generated

The Unbind_req is generated from a Local Device wishing to remove a Binding Table entry for the source and destination addresses contained as parameters. The destination addressing on this command SHALL be unicast only and the destination address SHALL be that of the SrcAddress.

2.4.3.2.3.2 Effect on Receipt

On receipt of a broadcast Unbind request the stack SHALL drop the message and no further processing SHALL be done. The Remote Device SHALL evaluate whether this request is supported. If the request is not supported, a Status of NOT_SUPPORTED SHALL be returned. If the request is supported, the Remote Device SHALL remove a Binding Table entry based on the parameters supplied in the Unbind_req. If a Binding Table entry for the SrcAddress, SrcEndp, ClusterID, DstAddress, DstEndp contained as parameters does not exist, the Remote Device SHALL respond with

NO_ENTRY. Otherwise, the Remote Device SHALL delete the indicated Binding Table entry and respond with SUCCESS.

2.4.3.2.4 Bind_Register_req – DEPRECATED

2.4.3.2.5 Replace_Device_req – DEPRECATED

2.4.3.2.6 Store_Bkup_Bind_Entry_req – DEPRECATED

2.4.3.2.7 Remove_Bkup_Bind_Entry_req – DEPRECATED

2.4.3.2.8 Backup_Bind_Table_req – DEPRECATED

2.4.3.2.9 Recover_Bind_Table_req – DEPRECATED

2.4.3.2.10 Backup_Source_Bind_req – DEPRECATED

2.4.3.2.11 Recover_Source_Bind_req – DEPRECATED

2.4.3.2.12 Clear_All_Bindings_req

The Clear_All_Bindings_req command (Cluster = 0x002b) SHALL be formatted as described in Figure 2-30. Any device on the network can send this command subject to the same Restricted Mode processing rules that apply to other commands manipulating the binding table.

Octets: Varies
TLVs

Figure 2-30. Format of the Clear_All_Bindings_req

The following TLVs SHALL be present in the message:

- Clear All Bindings Req EUI64 TLV

2.4.3.2.12.1 Local TLVs

2.4.3.2.12.2 Clear All Bindings Req EUI64 TLV (ID=0)

The format of the Clear All Bindings Req EUI64 TLV SHALL be as formatted in Figure 2-31.

Octets: 1	**8**	**…**
EUI64 Count	EUI64	…

Figure 2-31. Format of the Clear All Bindings Req EUI64 TLV

The fields of the Clear All Bindings Req EUI64 TLV are defined in Table 2-57.

Table 2-57. Fields of the Clear All Bindings Req EUI64 TLV

Name	Type	Valid Range	Description
EUI64 Count	Integer	0x00 – 0xFF	The number of EUI64 fields within the TLV. NOTE: The Maximum Transmission Unit (MTU) of the underlying message will limit the maximum range of this field.
EUI64	EUI64	0x0000000000000000 – 0xFFFFFFFFFFFFFFFF	An EUI64 that SHALL trigger corresponding bindings to be deleted.

2.4.3.2.12.3 When Generated

This is generated by a remote device that wants to clear all the bindings of the local device, for example to clear the application configuration without resetting the device to its factory defaults and causing it to drop off the network. This command SHALL be sent via unicast.

2.4.3.2.12.4 Effect on Receipt

The receiver SHALL do the following:

1) If the command was broadcast, the command SHALL be dropped and no further processing SHALL be done.
2) Perform TLV processing rules as described in Annex I (General TLV Processing section).
3) If the command does not include a Clear All Bindings Req EUI64 TLV in the message, then it SHALL be rejected.
    a) A ZDO Clear_All_Bindings_rsp SHALL be generated with a status of INV_REQUESTTYPE. No further processing SHALL be done to clear the application configuration without resetting the device to its factory defaults and causing it to drop off the network.
4) For each EUI64 in the Clear All Bindings Req EUI64 TLV search the Binding Table and delete any binding that matches that EUI64. If the Wildcard EUI64 of 0xFFFFFFFFFFFFFFFF is used then all bindings on the local device SHALL be deleted.
5) Generate a ZDO Clear_All_Bindings_rsp containing a Status Field.
    a) Set the status to SUCCESS

2.4.3.3 Network Management Client Services

Table 2-58 lists the commands supported by Device Profile: Network Management Client Services. Each of these primitives will be discussed in the following sections.

Table 2-58. Network Management Client Services Commands

Network Management Client Services	Cluster ID	Client Transmission	Server Processing	Restricted Command
Mgmt_NWK_Disc_req	0x0030	Deprecated	Deprecated	-
Mgmt_Lqi_req	0x0031	O	M	No
Mgmt_Rtg_req	0x0032	O	M	No
Mgmt_Bind_req	0x0033	O	M	No

Network Management Client Services	Cluster ID	Client Transmission	Server Processing	Restricted Command
Mgmt_Leave_req	0x0034	O	M	**Yes**
Mgmt_Direct_Join_req	0x0035	Deprecated	Deprecated	-
Mgmt_Permit_Joining_req	0x0036	O	M	No
Mgmt_Cache_req	0x0037	Deprecated	Deprecated	-
Mgmt_NWK_Update_req	0x0038	O	O	No
Mgmt_NWK_Enhanced_Update_req	0x0039	O	O	No
Mgmt_NWK_IEEE_Joining_List_req	0x003a	O	O	No
Reserved	0x003b	-	-	-
Mgmt_NWK_Beacon_Survey_req	0x003c	O	M*	No

* The Mgmt_NWK_Beacon_Survey_req server processing is mandatory for End Devices and optional for routers.

2.4.3.3.1 Mgmt_NWK_Disc_req – DEPRECATED

2.4.3.3.2 Mgmt_Lqi_req

The Mgmt_Lqi_req command (ClusterID=0x0031) SHALL be formatted as illustrated in Figure 2-32.

Octets: 1
StartIndex

Figure 2-32. Format of the Mgmt_Lqi_req Command Frame

Table 2-59 specifies the fields for the Mgmt_NWK_Disc_req command frame.

Table 2-59. Fields of the Mgmt_Lqi_req Command

Name	Type	Valid Range	Description
StartIndex	Integer	0x00 – 0xff	Starting Index for the requested elements of the Neighbor Table.

2.4.3.3.2.1 When Generated

The Mgmt_Lqi_req is generated from a Local Device wishing to obtain a neighbor list for the Remote Device along with associated LQA values to each neighbor. The destination addressing on this command SHALL be unicast only. It MAY be sent to a coordinator, router, or end device.

2.4.3.3.2.2 Effect on Receipt

Upon receipt, a Remote Device (Zigbee Router or Zigbee Coordinator) SHALL retrieve the entries of the neighbor table and associated LQA values via the NLME-GET.request primitive (for the *nwkNeighborTable* attribute) and report the resulting neighbor table (obtained via the NLME-GET.confirm primitive) via the Mgmt_Lqi_rsp command.[2]

Prior to Revision 21 of this specification, server processing of this command was optional. Additionally end devices were not required to support the command. As a result some devices MAY return NOT_SUPPORTED. For R22 and beyond, all devices SHALL support this command.

Prior to Revision 23 of this specification, the LQI value was returned, which might have exhibited more platform-specific behavior.

If this command is not supported in the Remote Device, the return status provided with the Mgmt_Lqi_rsp SHALL be NOT_SUPPORTED. If the neighbor table was obtained successfully, the Mgmt_Lqi_rsp command SHALL contain a status of SUCCESS and the neighbor table SHALL be reported, beginning with the element in the list enumerated as StartIndex. If the neighbor table was not obtained successfully, the Mgmt_Lqi_rsp command SHALL contain the error code reported in the NLME-GET.confirm primitive.

2.4.3.3.3 Mgmt_Rtg_req

The Mgmt_Rtg_req command (ClusterID=0x0032) SHALL be formatted as illustrated in Figure 2-33.

Octets: 1
StartIndex

Figure 2-33. Format of the Mgmt_Rtg_req Command Frame

Table 2-60 specifies the fields for the Mgmt_Rtg_req command frame.

Table 2-60. Fields of the Mgmt_Rtg_req Command

Name	Type	Valid Range	Description
StartIndex	Integer	0x00-0xff	Starting Index for the requested elements of the Routing Table.

2.4.3.3.3.1 When Generated

The Mgmt_Rtg_req is generated from a Local Device wishing to retrieve the contents of the Routing Table from the Remote Device. The destination addressing on this command SHALL be unicast only and the destination address SHALL be that of the Zigbee Router or Zigbee Coordinator.

[2] CCB 2265

2.4.3.3.3.2 **Effect on Receipt**

Upon receipt, a Remote Device (Zigbee Coordinator or Zigbee Router) SHALL retrieve the entries of the routing table from the NWK layer via the NLME-GET.request primitive (for the *nwkRouteTable* attribute) and report the resulting routing table (obtained via the NLME-GET.confirm primitive) via the Mgmt_Rtg_rsp command.

If the Remote Device does not support this optional management request, it SHALL return a Status of NOT_SUPPORTED. If the routing table was obtained successfully, the Mgmt_Rtg_req command SHALL contain a status of SUCCESS and the routing table SHALL be reported, beginning with the element in the list enumerated as StartIndex. If the routing table was not obtained successfully, the Mgmt_Rtg_rsp command SHALL contain the error code reported in the NLME-GET.confirm primitive.

2.4.3.3.4 Mgmt_Bind_req

The Mgmt_Bind_req command (ClusterID=0x0033) SHALL be formatted as illustrated in Figure 2-34.

Octets: 1
StartIndex

Figure 2-34. Format of the Mgmt_Bind_req Command Frame

Table 2-61 specifies the fields for the Mgmt_Bind_req command frame.

Table 2-61. Fields of the Mgmt_Bind_req Command

Name	Type	Valid Range	Description
StartIndex	Integer	0x00 – 0xff	Starting Index for the requested elements of the Binding Table.

2.4.3.3.4.1 **When Generated**

The Mgmt_Bind_req is generated from a Local Device wishing to retrieve the contents of the Binding Table from the Remote Device. The destination addressing on this command SHALL be unicast only and the destination address SHALL be that of a source device holding its own binding table.

2.4.3.3.4.2 **Effect on Receipt**

Upon receipt, a Remote Device SHALL retrieve the entries of the binding table from the APS sub-layer via the APSME-GET.request primitive (for the *apsBindingTable* attribute) and report the resulting binding table (obtained via the APSME-GET.confirm primitive) via the Mgmt_Bind_rsp command.

If the Remote Device does not support this optional management request, it SHALL return a status of NOT_SUPPORTED. If the binding table was obtained successfully, the Mgmt_Bind_rsp command SHALL contain a status of SUCCESS and the binding table SHALL be reported, beginning with the element in the list enumerated as StartIndex. If the binding table is empty, the Mgmt_Bind_rsp SHALL return SUCCESS, set the fields BindingTable Entries = Start Index = BindingTable ListCount = 0x00 and not include the BindingTable List field. If the binding table was not obtained successfully, the Mgmt_Bind_rsp command SHALL contain the error code reported in the APSME-GET.confirm primitive.

2.4.3.3.5 Mgmt_Leave_req

The Mgmt_Leave_req command (ClusterID=0x0034) SHALL be formatted as illustrated in Figure 2-35.

Bits: 64	6	1	1
Device Address	Reserved	Remove Children	Rejoin

Figure 2-35. Format of the Mgmt_Leave_req Command Frame

Table 2-62 specifies the fields for the Mgmt_Leave_req command frame.

Table 2-62. Fields of the Mgmt_Leave_req Command

Name	Type	Valid Range	Description
DeviceAddress	Device Address	An extended 64-bit, IEEE address	See section 3.2.2.18 for details on the Device Address parameter within NLME-LEAVE.request. For DeviceAddress of NULL, a value of 0x0000000000000000 SHALL be used.
Remove Children	Bit	0 or 1	This field has a value of 1 if the device being asked to leave the network is also being asked to remove its child devices, if any. Otherwise, it has a value of 0.
Rejoin	Bit	0 or 1	This field has a value of 1 if the device being asked to leave from the current parent is requested to rejoin the network. Otherwise, it has a value of 0.

2.4.3.3.5.1 When Generated

The Mgmt_Leave_req is generated from a Local Device requesting that a Remote Device leave the network or to request that another device leave the network. The Mgmt_Leave_req is generated by a management application which directs the request to a Remote Device where the NLME-LEAVE.request is to be executed using the parameter supplied by Mgmt_Leave_req.

2.4.3.3.5.2 Effect on Receipt

Upon receipt, the remote device SHALL process the leave request by executing the procedure in section 3.6.1.11.3.1. If the leave request was validated and accepted, and the DeviceAddress in the request is equal to the local device's EUI64, then the receiving device SHALL generate the NLME-LEAVE.request to disassociate from the currently associated network. The NLME-LEAVE.request SHALL have the DeviceAddress parameter set to the local device's *nwkIeeeAddress* from the NIB, the RemoveChildren SHALL be set to FALSE, and the Rejoin parameter SHALL be set to FALSE.

The results of the leave attempt SHALL be reported back to the local device via the Mgmt_Leave_rsp command. If the request was for the local device, then the Mgmt_Leave_rsp SHALL be sent prior to leaving the network.

Versions of this specification prior to Revision 21 did not mandate the requirement to support this command. Therefore if the remote device did not support this optional management request, it would return a status of NOT_SUPPORTED. All devices certified against version 21 and later are now required to support this command.

If the leave attempt was executed successfully, the Mgmt_Leave_rsp command SHALL contain a status of SUCCESS. If the leave attempt was not executed successfully, the Mgmt_Leave_rsp command SHALL contain the error code reported in the NLME-LEAVE.confirm primitive.

2.4.3.3.6 Mgmt_Direct_Join_req – DEPRECATED

2.4.3.3.7 Mgmt_Permit_Joining_req

The Mgmt_Permit_Joining_req command (ClusterID=0x0036) SHALL be formatted as illustrated in Figure 2-36.

Octets: 1	1	Variable
PermitDuration	TC_Significance	TLV Data

Figure 2-36. Format of the Mgmt_Permit_Joining_req Command Frame

Table 2-63 specifies the fields of the Mgmt_Permit_Joining_req command frame.

Table 2-63. Fields of the Mgmt_Permit_Joining_req Command

Name	Type	Valid Range	Description
PermitDuration	Integer	0x00 – 0xfe	See section 3.2.2.7 for details on the PermitDuration parameter within NLME-PERMIT-JOINING.request.
TC_Significance	Boolean Integer	0x00 – 0x01	This field SHALL always have a value of 1, indicating a request to change the Trust Center policy. If a frame is received with a value of 0, it shall be treated as having a value of 1.
TLV Data	Variable	Variable	This is a concatenated list of TLVs. This field was added in Revision 23.

2.4.3.3.7.1 When Generated

The Mgmt_Permit_Joining_req is generated from a Local Device requesting a change to the network's advertisement of its status, such as permitting joining. The Mgmt_Permit_Joining_req is generated by a management application or commissioning tool which directs the request to a remote device(s). Additionally, if the remote device is the Trust Center and TC_Significance is set to 1, this command is a request to change the Trust Center's policy to allow new devices to join. The Trust Center has the ultimate decision over whether this request will be accepted. The addressing MAY be unicast or 'broadcast to all routers and coordinator.

Trust Centers are the only devices allowed to update the Zigbee Beacon Appendix data advertised by the network in the IEEE Std 802.15.4 beacons. The network wide Beacon Appendix data is stored in the NIB value *nwkNetworkWideBeaconAppendixTLVs*.

The Trust Center can modify the nwkNetworkWideBeaconAppendixTLVs of all routers by setting data in the Beacon Appendix Encapsulation Global TLV. At a minimum the Trust Center SHALL always include the Beacon Appendix Encapsulation Global TLV as a TLV in the TLV Data field of a Mgmt_Permit_Joining_req. This is regardless of the value it sets for the PermitDuration field. Inside the Beacon Appendix Encapsulation Global TLV SHALL be the following TLVs:

- Supported Key Negotiation Methods Global TLV
- Fragmentation Parameter Global TLV

The Trust Center can include additional Global TLVs in the encapsulation TLV. Local TLVs SHALL NOT be stored in the Beacon Appendix Encapsulation Global TLV. The *nwkNetworkWideBeaconAppendixTLVs* SHALL always be set in its entirety by the Beacon Appendix Encapsulation Global TLV and SHALL NOT be appended to. The *nwkNetworkWideBeaconAppendixTLVs* NIB value SHALL NOT be persisted across reboots.

Additional local or global TLVs MAY be included in the TLV Data field of the Mgmt_Permit_Joining_req alongside the Beacon Appendix Encapsulation Global TLV. These TLVs do not change the state of the *nwkNetworkWideBeaconAppendixGlobalTLVs*.

Non-Trust Center devices are not allowed to change the network wide Beacon Appendix data advertised by the network, only the permit joining duration. Non-Trust Center devices initiating this message SHALL not include the Beacon Appendix Encapsulation Global TLV. They MAY include other TLVs in the TLV Data field of the Mgmt_Permit_Joining_req.

2.4.3.3.7.2 Effect on Receipt

Upon receipt, the remote device(s) SHALL issue the NLME-PERMIT-JOINING.request primitive using the PermitDuration parameter supplied with the Mgmt_Permit_Joining_req command. If the PermitDuration parameter is not equal to zero or 0xFF, the parameter is a number of seconds and joining is permitted until it counts down to zero, after which time, joining is not permitted. If the PermitDuration is set to zero, joining is not permitted. Versions of this specification prior to Revision 21 allowed a value of 0xFF to be interpreted as 'forever'. Version 21 and later do not allow this. All devices conforming to this specification SHALL interpret 0xFF as 0xFE Devices that wish to extend the PermitDuration beyond 0xFE seconds SHALL periodically re-send the Mgmt_Permit_Joining_req.

If a second Mgmt_Permit_Joining_req is received while the previous one is still counting down, it will supersede the previous request.

A value of zero for the TC_Significance field has been deprecated. The field SHALL always be included in the message and all received frames SHALL be treated as though set to 1, regardless of the actual received value. In other words, all Mgmt_Permit_Joining_req SHALL be treated as a request to change the TC Policy.

If the remote device is the Trust Center the Trust Center authorization policy MAY be affected. Whether the Trust Center accepts a change in its authorization policy is dependent upon its Trust Center policies. A Trust Center device receiving a Mgmt_Permit_Joining_req SHALL execute the procedure in section 4.7.3.4 to determine if the request is permitted. If the operation was not permitted, the status code of INV_REQUESTTYPE SHALL be set. If the operation was allowed, the status code of SUCCESS SHALL be set.

If the Mgmt_Permit_Joining_req primitive was received as a unicast, the results of the NLME-PERMIT-JOINING.request SHALL be reported back to the local device via the Mgmt_Permit_Joining_rsp command. If the command was received as a broadcast, no response SHALL be sent back.

Prior to Revision 23 the TLV Data was never present. With Revision 23 and beyond, the TLV Data field can be present depending on whether the message was initiated by a Revision 23 device. Devices prior to Revision 23 SHALL ignore the TLV Data field on receipt and will never transmit the message with this field present.

If the Beacon Appendix Encapsulation Global TLV is present the receiver SHALL store all Global TLVs from the TLV Data in the *nwkNetworkWideBeaconPayloadTLVs* of the NIB, the Beacon Appendix Encapsulation Global TLV container SHALL not be stored. If Local TLVs are stored inside the Beacon Appendix Encapsulation TLV they SHALL be discarded and not stored in the nwkNetworkWideBeaconPayloadTLVs. If the Beacon Appendix Encapsulation Global TLV Data has no data inside it, the receiver SHALL clear the contents of the *nwkNetworkWideBeaconPayloadTLVs* of the NIB .If the Beacon Appendix Encapsulation Global TLV is not present, then no changes are made to the contents of the *nwkNetworkWideBeaconPayloadTLVs* of the NIB.

2.4.3.3.8 Mgmt_Cache_req – DEPRECATED

2.4.3.3.9 Mgmt_NWK_Update_req

This command only supports the 2.4 GHz channel list. For other channels, see the Mgmt_NWK_Enhanced_Update_req.

The Mgmt_NWK_Update_req command (ClusterID=0x0038) SHALL be formatted as illustrated in Figure 2-37.

Octets: 4	1	0/1	0/1	0/2
ScanChannels	ScanDuration	ScanCount	*nwkUpdateId*	*nwkManagerAddr*

Figure 2-37. Fields of the Mgmt_NWK_Update_req Command Frame

Table 2-64 specifies the fields of the Mgmt_NWK_Update_req command frame.

Table 2-64. Fields of the Mgmt_NWK_Update_req Command

Name	Type	Valid Range	Description
ScanChannels	Bitmap	32-bit field	See Table 3-7 for details on the 32-bit field structure..
ScanDuration	Integer	0x00 – 0x05 or 0xfe or 0xff	A value used to calculate the length of time to spend scanning each channel. The time spent scanning each channel is (aBaseSuperframeDuration * (2^n + 1)) symbols, where n is the value of the ScanDuration parameter. For more information on MAC sub-layer scanning (see [B1]). If ScanDuration has a value of 0xfe this is a request for channel change. If ScanDuration has a value of 0xff this is a request to change the *apsChannelMaskList* and *nwkManagerAddr* attributes.
ScanCount	Integer	0x00 – 0x01	This field represents the number of energy scans to be conducted and reported. This field SHALL be present only if the ScanDuration is within the range of 0x00 to 0x05.
nwkUpdateId	Integer	0x00 – 0xFF	The value of the *nwkUpdateId* contained in this request. This value is set by the Network Channel Manager prior to sending the message. This field SHALL only be present of the ScanDuration is 0xfe or 0xff. If the ScanDuration is 0xff, then the value in the *nwkUpdateID* SHALL be ignored.
nwkManagerAddr	Device Address	16-bit NWK address	This field SHALL be present only if the ScanDuration is set to 0xff, and, where present, indicates the NWK address for the device with the Network Manager bit set in its Node Descriptor.

2.4.3.3.9.1 When Generated

This command is provided to allow updating of network configuration parameters or to request information from devices on network conditions in the local operating environment. The destination addressing on this primitive SHALL be unicast or broadcast to all devices for which macRxOnWhenIdle = TRUE.

2.4.3.3.9.2 Effect on Receipt

This section applies to both Mgmt_NWK_ Update_req and Mgmt_NWK_Enhanced_Update_req.

If Mgmt_NWK_Enhanced_Update_req is received and the server for it is not present, the device SHALL respond with NOT_SUPPORTED.

This command can cause the remote device to update its channel mask and network manager address, perform a channel change, or execute a channel scan. Processing is as follows.

1) If the received message is Mgmt_NWK_Update_req, the local device SHALL construct a ChannelListStructure for page 0 from the ScanChannels bitmap.
    a) Continue processing.
2) If the received message is Mgmt_NWK_Enhanced_Update_req, the local device SHALL construct a ChannelListStructure from the ScanChannelsListStructure.
    a) Continue processing.
3) If the ScanDuration parameter is equal to 0xfe, the message is a command to change channels. The device SHALL do the following.
    a) If the nwkNextChannelChange value in the NIB is non-zero, do the following. Compare the channel to change received over the air to the value in the NIB. If the values do not match, do the following:
        i) Follow the Error Response procedure setting the status to NOT_AUTHORIZED.
        ii) The request SHALL be dropped and no more processing SHALL take place.
    b) If there is more than 1 channel indicated in the ScanChannels bitmap (if the message is Mgmt_NWK_Update_req) or in the ScanChannelsListStructure (if the message is Mgmt_NWK_Enhanced_Update_req), then this is an invalid request. Do the following:
        i) Follow the Error Response procedure setting the status to INV_REQUESTTYPE.
        ii) The request SHALL be dropped and no more processing SHALL take place.
    c) The receiving device SHALL determine if the channel is one within the range of all supported channels.
        i) Examine the SupportedChannels element for each entry in the nwkMacInterfaceTable, and determine if there is a match within the received ScanChannels bitmap or ScanChannelsListStructure.
        ii) If no match is found, do the following:
            (1) Follow the Error Response procedure setting the status to INV_REQUESTTYPE.
            (2) The request SHALL be dropped and no more processing SHALL take place.
        iii) If a match is found, perform a channel change.
            (1) Execute a MLME-SET.request for the PIB value phyCurrentPage.
            (2) Execute a MLME-SET.request for the PIB value phyCurrentChannel.
            (3) No further processing SHALL be done.
4) If the ScanDuration parameter is equal to 0xff, the command provides a new apsChannelMaskList along with a new nwkManagerAddr. The device SHALL do the following:
    a) If the *apsTrustCenterAddress* of the AIB is set to a value other than 0xFFFFFFFFFFFFFFFF (distributed security network) and the nwkManagerAddr in the request is not 0x0000, the request SHALL be dropped and no more processing SHALL be done.
    b) If the received command is Mgmt_NWK_Update_req, set the apsChannelMaskList in the AIB to the value of the ScanChannels bitmap in the request.
    c) If the received command is a Mgmt_NWK_Enhanced_Update_req, use the value of the ScanChannelsListStructure in the request , to update the apsChannelMaskList in the AIB.
    d) Execute an NMLE-SET.request setting the nwkManagerAddr in the NIB to the value of the nwkManagerAddr in the request.

e) No more processing shall be done.

5) If the ScanDuration parameter is between 0x00 and 0x05, it is a request to do a channel scan. The device SHALL do the following:

a) If the request was not unicast, the request SHALL be dropped and no more processing SHALL be done.

b) For each entry in the nwkMacInterfaceTable, examine the SupportedChannels element and determine if there is a match.

i) If no match is found, do the following:

(1) Follow the Error Response procedure setting in section 2.4.3.3.9.3.

(2) The request SHALL be dropped and no more processing SHALL be done.

c) If the request is a Mgmt_NWK_Enhanced_Update_req and the ScanChannelsListStructure includes more than one page, do the following:

i) Follow the Error Response procedure setting the status to INV_REQUESTTYPE.

ii) The request SHALL be dropped and no more processing SHALL be done.

d) If a match is found, perform an Energy Detect Scan on the requested channels. The following procedure SHALL be executed a number of times equal to the ScanCount.

i) Execute a MLME-SCAN.request as follows.

(a) ScanType SHALL be set to ENERGY.

(b) ScanChannels SHALL be set to the matching channels in the current page.

(c) ChannelPage SHALL be set to the current page.

(d) ScanDuration SHALL be set to the ScanDuration in the request.

ii) If the received message is a Mgmt_NWK_Update_req, on receipt of the MLME-SCAN.confirm, generate a Mgmt_NWK_Update_notify with the status of the MLME-SCAN.confirm.

iii) If the received message is a Mgmt_NWK_Enhanced_Update_req, on receipt of the MLME-SCAN.confirm, generate a Mgmt_NWK_Enhanced_Update_notify with the status of the MLME-SCAN.confirm..

6) If the ScanDuration is any other value, the device SHALL do the following.

a) Execute the Error Response Procedure setting the status to INV_REQUESTTYPE.

b) No further processing SHALL be done.

2.4.3.3.9.3 Error Response Procedure

If it is determined that the error response procedure SHALL be executed, the device SHALL do the following:

1) If the request was broadcast, no response SHALL be generated.

2) If the request was unicast, a response SHALL be generated as follows:

a) Set the status according to the result of the operation.

b) If the request was a Mgmt_NWK_Update_req, generate a Mgmt_NWK_Update_notify.

c) If the request was a Mgmt_NWK_Enhanced_Update_req, generate a Mgmt_NWK_Enhanced_Update_notify.

2.4.3.3.10 Mgmt_NWK_Enhanced_Update_req

The Mgmt_NWK_Enhanced_Update_req command (ClusterID=0x0039) SHALL be formatted as illustrated in Figure 2-38.

Variable	**1**	**0/1**	**0/1**	**0/2**	**0/1**
ScanChannel-ListStructure	ScanDuration	ScanCount	*nwkUpdateId*	*nwkManagerAddr*	ConfigurationBitmask

Figure 2-38. Fields of the Mgmt_NWK_Enhanced_Update_req

Table 2-65 specifies the fields of the Mgmt_NWK_Enhanced_Update_req command frame.

Table 2-65. Field Descriptions of the Mgmt_NWK_Enhanced_Update_req

Name	**Type**	**Valid Range**	**Description**
ScanChannelsListStructure	ChannelListStructure	Variable	The list of channels and pages over which the scan is to be done. For more information on the Channel List structure see section 3.2.2.2.1. If ScanDuration is in the range 0x00 to 0x05, this parameter SHALL be restricted to a single page.
ScanDuration	Integer	0x00 – 0x05 or 0xfe or 0xff	A value used to calculate the length of time to spend scanning each channel. The time spent scanning each channel is (aBaseSuperframeDuration * (2^n + 1)) symbols, where n is the value of the ScanDuration parameter. For more information on MAC sub-layer scanning (see [B1]). If ScanDuration has a value of 0xfe this is a request for channel change. If ScanDuration has a value of 0xff this is a request to change the *apsChannelMaskList* and *nwkManagerAddr* attributes.
ScanCount	Integer	0x00 – 0x01	This field represents the number of energy scans to be conducted and reported. This field SHALL be present only if the ScanDuration is within the range of 0x00 to 0x05.
nwkUpdateId	Integer	0x00 – 0xFF	The value of the *nwkUpdateId* contained in this request. This value is set by the Network Channel Manager prior to sending the message. This field SHALL only be present if the ScanDuration is 0xfe or 0xff.

Name	Type	Valid Range	Description
			If the ScanDuration is 0xff, then the value in the *nwkUpdateID* SHALL be ignored.
nwkManagerAddr	Device Address	16-bit NWK address	This field SHALL be present only if the ScanDuration is set to 0xff, and, where present, indicates the NWK address for the device with the Network Manager bit set in its Node Descriptor.
ConfigurationBitmask			Defined in defined in section 2.4.3.3.12. The configurationBitmask must be added to the end of the list of parameters. This octet may or may not be present. If not present then assumption should be that it is enhanced active scan. If present then the configuration bitmask shall indicate the type of scan required.

2.4.3.3.10.1 When Generated

This command is provided to allow updating of network configuration parameters or to request information from devices on network conditions in the local operating environment. The destination addressing on this primitive SHALL be unicast or broadcast to all devices for which macRxOnWhenIdle = TRUE.

2.4.3.3.10.2 Effect on Receipt

Follow the procedure in .

2.4.3.3.11 Mgmt_NWK_IEEE_Joining_List_req

The Mgmt_NWK_IEEE_Joining_List_req command is provided as a mechanism to obtain the list of IEEE addresses that are EXPECTED to be joining the network. This allows the local router to filter Enhanced Beacon Requests and only respond to the devices that are joining.

The Mgmt_NWK_IEEE_Joining_List_req (Cluster ID 0x003A) command SHALL be formatted as illustrated in Figure 2-39.

Octets: 1
StartIndex

Figure 2-39. Fields of the Mgmt_NWK_IEEE_Joining_List_req

Table 2-66 describes the fields of the Mgmt_NWK_IEEE_Joining_List_req command.

Table 2-66. Field Descriptions of the Mgmt_NWK_IEEE_Joining_List_req

Name	Type	Valid Range	Description
StartIndex	Integer	0x00 – 0xFF	The starting index into the receiving device's nwkIeeeJoiningList that SHALL be sent back.

2.4.3.3.11.1 When Generated

The Mgmt_NWK_IEEE_Joining_List_req is generated from a local device requesting to get the mibJoinPolicyTable after being authenticated on the network.

2.4.3.3.11.2 Effect on Receipt

This command was introduced in R22 of this specification. It is mandatory for all Coordinator and Router devices to implement this going forward but older stack versions SHALL return a ZDO Status of NOT_SUPPORTED upon receipt of this command.

The following procedure SHALL be executed upon receipt of this command.

1) If this request is broadcast, the message shall be dropped and no further processing SHALL be done.
2) The device SHALL obtain the *mibJoiningIeeeList* and *mibJoiningPolicy* from one of its currently enabled MAC Interfaces.
    a) Examine the *nwkMacInterfaceTable* and obtain an entry where State is set to ENABLED.
    b) Execute an MLME-GET.request for *mibJoiningIeeeList* and *mibJoiningPolicy*.
3) If the mibIeeeJoiningList is empty, then a Mgmt_NWK_IEEE_Joining_List_rsp SHALL be generated as follows.
    a) Status SHALL be set to SUCCESS.
    b) JoiningPolicy SHALL be set to the value of the *mibJoiningPolicy*.
    c) IeeeJoiningListTotal SHALL be set to 0.
    d) Unicast the response back to the sender of the Mgmt_NWK_IEEE_Joining_List_req.
    e) No further processing SHALL be done.
4) The device SHALL examine the StartIndex field and determine if it is less than the length of the mibJoiningIeeeList. If it is not, it SHALL do the following:
    a) A Mgmt_NWK_IEEE_Joining_List_rsp SHALL be generated with a Status value of INVALID_INDEX. No other fields shall be appended.
    b) Unicast the response back to the sender of the Mgmt_NWK_IEEE_Joining_List_req.
    c) No further processing SHALL be done.
5) The device SHALL generate a Mgmt_NWK_IEEE_Joining_List_rsp.

a) Set the Status value to SUCCESS.

b) Set the JoiningPolicy in the response to the previously obtain value of mibJoiningPolicy.

c) Set the StartIndex of the response packet equal to the value of the StartIndex in the request packet.

d) Copy complete IEEE addresses from the mibJoiningIeeeList to the IeeeJoiningList, from the Start Index, filling the payload of the packet up to the MTU.

e) Set the IeeeJoiningListTotal to the number of complete entries that were copied.

f) Unicast the response back to the sender of the Mgmt_NWK_IEEE_Joining_List_req.

2.4.3.3.12 Mgmt_NWK_Beacon_Survey_req

This command can be used by a remote device to survey the end devices to determine how many potential parents they have access to. The Mgmt_NWK_Beacon_Survey_req command (cluster ID 0x003c) SHALL be formatted as described in Figure 2-40.

Octets: Varies
TLVs

Figure 2-40. Format of the Mgmt_NWK_Beacon_Survey_req

Table 2-67 describes the fields of the Mgmt_NWK_Beacon_Survey_req command.

Table 2-67. Fields of the Mgmt_NWK_Beacon_Survey_req

Name	Type	Valid Range	Description
TLVs	TLV	Varies	The following TLVs SHALL be included in the Mgmt_NWK_Beacon_Survey_req: • Beacon Survey Configuration TLV

2.4.3.3.12.1 Beacon Survey Configuration TLV

The Beacon Survey Configuration TLV (ID=0) is variable in length and contains information about the channels and scan configuration used when performing a beacon survey. The format is listed in Figure 2-41.

Octets: Variable	1
ScanChannelListStructure	ConfigurationBitmask

Figure 2-41. Format of the Beacon Survey Configuration TLV

2.4.3.3.12.1.1 ScanChannelsListStructure

Name	Type	Valid Range	Description
ScanChan-nelsListStructure	Channel-ListStruc-ture	Variable	The list of channels and pages over which the scan is to be done. For more information on the Channel List structure see section 3.2.2.2.1.

2.4.3.3.12.1.2 Configuration Bitmask

This field indicates parameters of the Mgmt_NWK_Beacon_Survey_req. The Configuration bitmask enumerated values are specified in Table 2-68.

Table 2-68. Configuration Bitmask Values

Bit	Name	Description
0	Active or Enhanced Scan	This bit determines whether to do an Active Scan or Enhanced Active Scan. When the bit is set to 1 it indicates an Enhanced Active Scan. And in case of Enhanced Active scan EBR shall be sent with EPID filter instead of PJOIN filter.
1 – 7	Reserved	-

2.4.3.3.12.2 When Generated

This is generated by a remote device that wants to learn how many potential parents a Zigbee End Device has. The message SHALL be sent as a unicast to a single target device.

2.4.3.3.12.3 Effect on Receipt

The processing of the Mgmt_NWK_Beacon_Survey_req SHALL be done as follows:

1) If the command was broadcast it SHALL be dropped and no further processing SHALL be done.
2) If the command does not contain the mandatory TLVs listed in Figure 2-40. Format of the Mgmt_NWK_Beacon_Survey_req

Table 2-67 describes the fields of the Mgmt_NWK_Beacon_Survey_req command.

3) Table 2-67 then a Mgmt_Beacon_Survey_rsp SHALL be generated with a status of MISSING_TLV and no further processing SHALL be done.
4) If the command is received by a coordinator, the coordinator SHALL reject the command. The coordinator does not perform rejoins and thus does not need to be surveyed in this manner.
   a) The coordinator shall construct a Mgmt_NWK_Beacon_Survey_rsp with a status field value of NOT_PERMITTED and no further payload fields. It SHALL unicast the response back to the sender and no further processing SHALL be done.
5) Construct a Beacon Survey Results TLV with all sub-fields set to 0.
6) Construct a Potential Parent TLV.
   a) If the device is an End Device, set the Current parent value to the Short Address of its parent.
   b) If the device is a Router, set the current parent to 0xFFFF.
7) If the Configuration field in the Beacon Survey Configuration TLV indicates Enhanced Active Scan and the local device does not support ENHANCED_ACTIVE, then a Mgmt_Beacon_Survey_rsp SHALL be generated with a status of INV_REQUESTTYPE and no further processing SHALL be done.
8) Execute an MLME-SCAN.request with the following parameters:
   a) If the Configuration field in the Beacon Survey Configuration TLV indicates Enhanced Active Scan, set the ScanType to ENHANCED_ACTIVE. Otherwise set to ACTIVE.
   b) ScanChannels set to the list of channels contained in the Beacon Survey Configuration TLV.
9) Upon receipt of the MLME-BEACON-NOTIFY.indication process the beacons as follows:
   a) Increment the Total Beacons Field by 1.
   b) For each beacon that has a Zigbee Beacon Payload and the Extended PAN ID field of that beacon payload is equal to the nwkExtendedPanId, do the following:
      i) Increment the On-Network Beacons field.
      ii) If the End Device Capacity of the Zigbee Beacon Payload is TRUE, increment the Potential Parent Beacons field by 1.
   c) If there is no Zigbee Beacon Payload or the Extended PAN ID does not match the nwkExtendedPanId, do the following:
      i) Increment the Other Network Beacons field by 1.

    d) Evaluate the beacon, potentially adding it to the Discovery Table (nwkDiscoveryTable).
    e) If any of the above values reach 255, they SHALL NOT wrap and be set to 255.
10) Add up to 5 devices into the Potential Parent TLV from the contents of the nwkDiscoveryTable. Update the Count of Potential Parents accordingly.
11) Generate a ZDO Mgmt_NWK_Beacon_Survey_rsp to the sender of the request with the following TLVs
    a) Beacon Survey Results TLV.
    b) Potential Parents TLV
    c) Pan ID Conflict Report Global TLV
        i) If the device is an End Device and does not support this NIB value, this TLV may be omitted.
        ii) Note: The nwkPanIdConflictCount value in the NIB SHALL NOT be reset to 0.
12) Discard the results stored in the *nwkDiscoveryTable*.

2.4.3.4 Security Client Services

Security Client Services allow devices to configure security policies, retrieve security policies, negotiate keys, and update security tokens. Table 2-69 lists the commands supported by the Device Profile related to Security Client services.

Table 2-69. Security Client Services Commands

Security Client Services	Cluster ID	Client Transmission	Server Processing	Restricted Command
Security_Start_Key_Negotiation_req	0x0040	O	O	No
Security_Retrieve_Authentication_Token_req	0x0041	O	O	No
Security_Get_Authentication_Level_req	0x0042	O	O	No
Security_Set_Configuration_req	0x0043	O	M	No
Security_Get_Configuration_req	0x0044	O	M	No
Security_Start_Key_Update_req	0x0045	O	M	No
Security_Decommission_req	0x0046	O	M	No
Security_Challenge_req	0x0047	M	M	No

2.4.3.4.1 Security_Start_Key_Negotiation_req

The Security_Start_Key_Negotiation_req command (0x0040) shall be formatted as illustrated in Figure 2-42. This command SHALL NOT be APS encrypted regardless of whether sent before or after the device joins the network.

This command SHALL be network encrypted if the device has a network key, i.e. it has joined the network earlier and wants to negotiate or renegotiate a new link key; otherwise, if it is used prior to joining the network, it SHALL NOT be network encrypted.

Octets: Variable
TLVs

Figure 2-42. Format of the Security_Start_Key_Negotiation_req Command

Table 2-70 describes the fields of the Security_Start_Key_Negotiation_req command.

Table 2-70. Fields of the Security_Start_Key_Negotiation_req Command

Name	Type	Valid Range	Description
TLVs	TLVs	Varies	A list of one or more TLVs. The following TLVs have specified behavior in this release of the specification: • Curve25519 Public Point TLV Other TLVs may be included.

2.4.3.4.1.1 Local TLVs

2.4.3.4.1.2 Curve25519 Public Point TLV (ID=0)

Figure 2-43 indicates the format of the Curve25519 Public Point TLV.

Octets: 8	32
Device EUI64	Public Point

Figure 2-43. Format of the Curve25519 Public Point TLV

Table 2-71 describes the fields of the Curve25519 Public Point TLV.

Table 2-71. Fields of the Curve25519 Public Point TLV

Field	Description
Device EUI64	This indicates the EUI64 of the device that generated the public point.
Public Point	The 32-byte Curve public point.

2.4.3.4.1.3 When Generated

The Security_Start_Key_Negotiation_req is generated from a local device that wants to start negotiation of an encryption key. Typically, this is used to negotiate a Trust Center Link Key during the joining process prior to becoming fully authorized on the network. However, it can be used after joining a network as well. Refer to section 4.6.3.5.

The security primitives for key negotiation are the APSME-KEY-NEGOTIATION primitives and are used by the stack to manage the process. See section 4.4.9 for more details. Their interaction with the over-the-air messages can be found in Figure 4-6.

When negotiating a Trust Center Link Key the device SHALL send at least the following TLV:

- Curve25519 Public Point TLV

It is EXPECTED that the sending device has already been told the selected Key Negotiation Protocol and selected Pre-Shared Secrets of the target device prior to sending this message. The sending device can learn the Supported Key Negotiation Methods in one of two possible ways: (1) in case of on-network key negotiation, the device sends first a Node Descriptor Request advertising its own supported key negotiation methods and the Node Descriptor Response will contain the selected Key Negotiation Protocol and selected Pre-Shared secret; (2) in case of off-network key negotiation, the Trust Center sends a Security Start Key Update Request with the selected Key Negotiation Protocol and selected Pre-Shared secret, after it has received the TLVs conveyed in a Network Commissioning request. If the sending device supports multiple mechanisms, via implementation-specific configuration it SHALL choose one that is supported by the target device.

2.4.3.4.1.4 Effect on receipt

The Device EUI64 within the Curve25519 Public Point TLV SHALL represent the EUI64 of the device that is requesting the key negotiation with the receiving device. The processing of the message SHALL be done as follows:

1. Execute the General TLV Processing Rules in Annex I
    a. If the outcome is to reject the message, do the following.
        i. If the message was broadcast, no response is generated.
        ii. If the message is unicast, a Security_Key_Negotiation_rsp SHALL be generated with a status as returned by the General TLV Processing rules Key Exchange. The response SHALL be sent back to the sender of the Security_Retrieve_Authentication_Token_req.
        iii. No further processing SHALL be done.
    b. Otherwise, continue processing.
2. If the Curve25519 Public Point TLV is not present, then a ZDO Security_Key_Negotiation_rsp SHALL be generated with a status of MISSING_TLV.
3. Generate an APSME-KEY-NEGOTIATION.indication with the following parameters:
    a. The RequestedKeyNegotiationMethod SHALL be set to the value conveyed in the Node_Desc_rsp Selected Key Negotiation Method TLV or Security_Start_Key_Update_req Selected Key Negotiation Method TLV.
    b. The PartnerLongAddress SHALL be set to the Device EUI64 within the Curve25519 Public Point TLV.
    c. The PublicPointData SHALL be set to the public point from the Curve25519 Public Point TLV.
    d. If the ZDO frame was contained within an APS Command Relay Message Downstream, then it SHALL do the following
        i. Set RelayCommand to TRUE
        ii. Set RelayLongAddress to the address of the Device that sent the Network Data frame.

2.4.3.4.2 Security_Retrieve_Authentication_Token_req

The Security_Retrieve_Authentication_Token_req command (0x0041) shall be formatted as illustrated in Figure 2-44. This command SHALL be APS encrypted.

Octets: Variable
TLVs

Figure 2-44. Format of the Security_Retrieve_Authentication_Token_req Command

Table 2-72 describes the fields of the Security_Start_Key_Negotiation_req command.

Table 2-72. Fields of the Security_Retrieve_Authentication_Token_req Command

Name	Type	Valid Range	Description
TLVs	TLVs	Varies	A list of one or more TLVs. The following TLVs have specified behavior in this release of the specification: • Authentication Token ID TLV Other TLVs may be included.

This command is used to retrieve a security token that can be used for future authentication exchanges. Security tokens could take multiple forms such as certificates, public keys, or symmetric passphrase. As of this Revision of this specification, only a symmetric passphrase is supported. The current use of this command is to obtain a new passphrase token. The passphrase token is intended to be good for the life of the device on that network. Previously, the device SHALL have been added to the keytable of the Trust Center during the APS update device. Once the device has obtained a new passphrase, replacing either a well-known pre-shared secret or one derived from an install code or passcode, it is locked down and not allowed to be replaced automatically. A Trust Center MAY administratively reset the device's security and thus allow it to join again and get a new token.

The passphrase used to join the network is intended to be used only once and then the device SHALL update it. The initial passphrase is either well-known (unauthenticated) or is the install code derived link key (authenticated). Once passphrase is updated it is never intended to be changed again for the life of the device on the network. The key negotiation leverages the passphrase and the devices need to avoid a circumstance where there is a passphrase mismatch, which could prevent the devices from ever successfully negotiating a symmetric link key again.

2.4.3.4.2.1 Local TLVs

2.4.3.4.2.2 Authentication Token ID TLV (ID=0)

The Authentication Token ID TLV is formatted as shown in Figure 2-45.

Octets: 1
TLV Type Tag ID

Figure 2-45. Authentication Token ID TLV

Table 2-73 describes the fields of the Authentication Token ID TLV.

Table 2-73. Requested Token ID TLV

Field	Description
TLV Type Tag ID	The Global TLV Type Tag ID being requested for an authentication token.

2.4.3.4.2.3 When Generated

This command is used to request a unique device specific authentication token that can be used for future key re-negotiation. This token can be used across a replacement of the Trust Center.

A device SHALL include the authentication token type that it supports by sending the Authentication Token ID TLV with the Global TLV Type Tag ID. The only supported authentication token in this specification is 128-bit Symmetric Passphrase Global TLV.

By sending the TLV Type Tag ID this potentially allows a future specification to use alternate tokens. For example, the Type Tag ID requested could be an operational certificate and the Trust Center could sign the ephemeral public key the joiner used during joining and then send it back to the device.

Authentication tokens are only updated with this command by a device requesting one from the Trust Center. This is not used for Partner Link Key Negotiation.

2.4.3.4.2.4 Effect on receipt

Upon receipt, a device that is not the Trust Center SHALL respond with a Security_Retrieve_Authentication_Token_rsp with a status of NOT_SUPPORTED and no further processing SHALL be done. If the received message is not APS encrypted, or it is a broadcast, then the message SHALL be dropped and no further processing SHALL be done.

Obtaining a security token of a specific type SHALL only be done once during join. The token is intended to be good for the life of the device on that network. In this Revision of the specification, only the 128-bit Symmetric Passphrase is a valid token type, but to allow for future security extensions, obtaining a security token of a different type may be permitted, based on the Trust Center policy. Previously, the device SHALL have been added to the key-table during the APS update device. Once the device has obtained a new passphrase, replacing either a well-known pre-shared secret or one derived from an install code, it is locked down and not allowed to be replaced automatically. A Trust Center MAY administratively reset the device's security and thus allow it to join again and get a new token.

The Trust Center SHALL perform the following:

1. Execute the General TLV Processing Rules in Annex I.
    a. If the outcome is to reject the message, do the following:
        i. If the message was broadcast, no response is generated.
        ii. If the message is unicast, a Security_Retrieve_Authentication_Token_rsp SHALL be generated with a status of INVALID_TLV. The response SHALL be sent back to the sender of the Security_Retrieve_Authentication_Token_req.
        iii. No further processing SHALL be done.
    b. Otherwise, continue processing.
2. The Trust Center SHALL search *apsDeviceKeyPairSet* table in the AIB for an entry that matches the EUI64 of the request.
    a. If none is found then a Security_Retrieve_Authentication_Token_rsp SHALL be generated to the requesting device with a status of NOT_PERMITTED, and no further processing SHALL be done.
    b. Otherwise, continue processing.
3. If the Authentication Token ID TLV is not present then the following steps SHALL be done.
    a. A Security_Retrieve_Authentication_Token_rsp SHALL be generated to the requesting device with a status of INVALID_TLV, and no further processing SHALL be done.
4. The Trust Center SHALL examine the TLV Tag ID in the Authentication Token ID TLV received in the message.
    a. If the TLV Tag ID is not 69, 128-bit Symmetric Passphrase Global TLV then a Security_Retrieve_Authentication_Token_rsp SHALL be generated to the requesting device with a status of INV_REQUESTTYPE, and no further processing SHALL be done.
5. The Trust Center SHALL examine the value of PassphraseUpdateAllowed for the entry of the apsDeviceKeyPairSet.
    a. If this value is set to FALSE then a Security_Retrieve_Authentication_Token_rsp SHALL be generated to the requesting device with a status of NOT_PERMITTED, and no further processing SHALL be done.
    b. Otherwise, continue processing.
6. The Trust Center SHALL generate a random 128-bit number with a cryptographically secure random number generator.
7. The Trust Center SHALL store the value as the Passphrase value for the associated entry of the *apsDeviceKeyPair* table AIB value.
8. The Trust Center SHALL construct a 128-bit Symmetric Passphrase Global TLV containing the value.
9. The Trust Center SHALL generate a Security_Retrieve_Authentication_Token_rsp to the sender of the request with a status of SUCCESS and the created TLV.
10. The Trust Center SHALL set the PassphraseUpdateAllowed value to FALSE for the associated entry of the *apsDeviceKeyPair* table AIB value.

2.4.3.4.3 Security_Get_Authentication_Level_req

This command allows a device to query the trust center about a 3rd party device to determine how it is authenticated on the network. This enables the querying device to determine if that 3rd party has the minimum required authentication level for application communication.

The Security_Get_Authentication_Level_req command (ClusterID=0x0042) shall be formatted as illustrated in Figure 2-46. It SHALL have APS encryption.

Octets: Variable
TLVs

Figure 2-46. Format of the Security_Get_Authentication_Level_req Command

Table 2-74 describes the fields of the Security_Get_Authentication_Level_req command.

Table 2-74 Fields of the Security_Get_Authentication_Level_req Command

Name	Type	Valid Range	Description
TLVs	TLVs	Varies	A list of one or more TLVs. The following TLVs have specified behavior in this release of the specification: • Target IEEE Address TLV Other TLVs may be included.

2.4.3.4.4.3.1 Local TLVs

2.4.3.4.4.3.2 Target IEEE Address TLV (ID=0)

The format of the Target IEEE Address TLV is shown in Figure 2-47.

Octets: 8
IEEEAddrOfInterest

Figure 2-47. Format of the Target IEEE Address TLV

Table 2-75 specifies the fields of the Target IEEE Address TLV.

Table 2-75. Fields of the Target IEEE Address TLV

Name	Type	Valid Range	Description
IEEEAddrOfInterest	64-bit IEEE address	Any	Extended address of the device whose security level is requested.

2.4.3.4.4.3.3 When Generated

The Security_Get_Authentication_Level_req is generated by the local device wishing to find out the authentication level of another device on the network. The command SHALL be unicast to the trust center. This command SHALL be APS encrypted.

2.4.3.4.4.3.4 Effect on Receipt

The following SHALL occur on the receipt of the Security_Get_Authentication_Level_req.

1. If the Security_Get_Authentication_Level_req command is broadcast it SHALL be dropped and no further processing SHALL be done.
2. If the Security_Get_Authentication_Level_req is not APS encrypted it SHALL be dropped and no further processing SHALL be done.
3. If the receiving device is NOT the Trust Center then a Security_Get_Authentication_Level_rsp SHALL be generated with a status of NOT_AUTHORIZED and no further processing SHALL be done.
4. Execute the General TLV Processing Rules in Annex I.4.7.
5. If the Target IEEE Address TLV is not present in the message the receiver SHALL generate a Security_Authentication_Level_rsp with a status of INV_REQUESTTYPE and no further processing SHALL be done.

6. The receiving device SHALL examine the *apsDeviceKeyPairSet* table of the AIB to find the entry matching the IEEEAddrOfInterest present in the Target IEEE Address TLV.
    a. If no matching entry is found then a Security_Get_Authentication_Level_rsp SHALL be generated with a status of NO_MATCH and no further processing SHALL be done.
7. If the IEEEAddrRemoteNode is the address of the Trust Center or 0xFFFFFFFFFFFFFFFF, the receiver SHALL generate a Security_Authentication_Level_rsp with a status of INV_REQUESTTYPE and no further processing SHALL be done.
8. The Device SHALL create a Security_Get_Authentication_Leve1_rsp with the following values:
    a. Set the status of response to SUCCESS
    b. Create a Device Authentication Level TLV
        a. Set the IEEEAddrRemoteNode of the response to the IEEEAddrOfInterest received in the request frame.
        b. From the matching entry in the apsDeviceKeyPairSet table set the InitialJoinMethod value in the Device Authentication Level TLV to the value of the InitialJoinAuthentication value from the AIB entry.
        c. From the matching entry in the apsDeviceKeyPairSet table set the ActiveLinkKeyType value in the Device Authentication Level TLV to the value of the PostJoinKeyUpdateMethod value from the AIB entry.

2.4.3.4.4 Security_Set_Configuration_req

The Security_Set_Configuration_req allows the Trust Center to change configuration options for a particular device.

The format of the message is in Figure 2-48. This command SHALL be APS encrypted when operating in a centralized security network. When operating in a distributed security network the command MAY be APS encrypted.

Octets: Varies
TLVs

Figure 2-48. Format of the Security_Set_Configuration_req Command

Table 2-76 specifies the fields of the Security_Set_Configuration_req command.

Table 2-76. Fields of the Security_Set_Configuration_req Command

Name	Type	Range	Description
TLVs	TLVs	Varies	A list of one or more TLVs. The following TLVs have specified behavior in this Revision of the specification: • Next PAN ID Global TLV • Next Channel Change Global TLV • Configuration Parameters Global TLV Other TLVs may be included.

The fields of the command Security_Set_Configuration_req are specified in Table 2-76. The following TLVs MAY be present:

- Next PAN ID Change Global TLV
- Next Channel Change Global TLV
- Configuration Parameters Global TLV

2.4.3.4.4.1 Local TLVs

There are no Local TLVs defined for this command.

2.4.3.4.4.2 When Generated

This is generated by the Trust Center when it wants to change configuration settings of the device. In distributed security networks, it MAY be generated by any device that wants to change the configuration settings of a remote device. In a distributed security network, it is permissible to send this command as a broadcast.

2.4.3.4.4.3 Effect on Receipt

When operating in a centralized security network, on receipt of a Security_Set_Configuration_req sent to the broadcast address, the device SHALL drop the message and no further processing SHALL be done.

When operating in a centralized security network, on receipt of a Security_Set_Configuration_req from a device that is not the Trust Center, the receiving device SHALL generate a Security_Set_Configuration_rsp with a status of NOT_AUTHORIZED. No further processing SHALL be done.

When operating in a distributed network, this command MAY be broadcast or unicast and MAY or MAY NOT be APS encrypted. The command is accepted in all those cases.

On receipt of a Security_Set_Configuration_req by the Trust Center, the Trust Center device SHALL generate a Security_Set_Configuration_rsp with a status of NOT_AUTHORIZED. No further processing SHALL be done.

Processing of the message SHALL be done as follows:

1. Execute the General TLV Processing rules in Annex I.
2. If the result of the processing indicates a failure, then do the following.
   a. If the command was unicast, the receiver SHALL transmit a Security_Set_Configuration_rsp to the sender with the status code as returned from the General TLV Processing rules.
   b. If the command was broadcast, no response is generated.
   c. For all cases when the TLV processing fails, no further processing SHALL be done.
3. Process the TLVs in the message as follows:
   a. Upon receipt of the Configuration Parameters Global TLV, the stack SHALL modify the value of the corresponding information base value as referenced in Table 4 36.
   b. Upon receipt of the Next PAN ID Global TLV, the stack SHALL modify the NIB value of the nwkNextPanId according to the value received in the TLV. Setting the nwkNextPanId to the broadcast PAN ID is allowed. It indicates that any PAN ID MAY be used as the next PAN ID.
   c. Upon receipt of the Next Channel Change Global TLV the stack SHALL modify the NIB value of the nwkNextChannelChange in the NIB according to the value received in the TLV if it matches one of the Supported Channels of an interface in the nwkMacInterfaceTable.

4. If no TLVs were processed in step 3, do the following:
   a. If the command was broadcast, no more processing SHALL take place.
   b. If the command was unicast, send a Security_Set_Configuration_rsp with a status MISSING_TLV to the sender of the request.

Note that an Overall Status of NOT_SUPPORTED for the Security_Set_Configuration_rsp is reserved for stacks prior to Revision 23 that do not understand the Security_Get_Configuration_req command at all.

2.4.3.4.5 Security_Get_Configuration_req

This command is used by a device to retrieve a remote device's security configuration. The Security_Get_Configuration_req command (cluster ID = 0x0044) is formatted as illustrated in Figure 2-49.

This command SHALL be APS encrypted in centralized security mode. It MAY be APS encrypted in distributed security mode.

Octets: 1	1	…
TLV Count	TLV ID	…

Figure 2-49. Format of the Security_Get_Configuration_req Command Frame

Table 2-77 specifies the fields of the Security_Get_Configuration_req command frame.

Table 2-77. Fields of the Security_Get_Configuration_req Command Frame

Name	Type	Valid Range	Description
TLV Count	Integer	0 – 255	The number of TLV IDs contained in the message. Note that the maximum value for this count will be dependent on the underlying maximum size of the message as allowed by fragmentation.
TLV ID	Integer	0 – 255	The ID of each TLV that is being requested.

2.4.3.4.5.1 When Generated

This is generated by a device that wants to retrieve the configuration of a remote device.

2.4.3.4.5.2 Effect on Receipt

If the command was broadcast it SHALL be rejected and silently dropped.

In a centralized security network, if the local device receives this command from a remote device that is not the Trust Center the command SHALL be rejected. The receiver SHALL generate a Security_Get_Configuration_rsp with a status of NOT_AUTHORIZED. No further processing SHALL be done.

The following processing SHALL be done.

1. Construct a Security_Get_Configuration_rsp command with a status of SUCCESS.
2. For each TLV ID listed in the message, the device SHALL determine if the TLV is known to the local device and has a value.
    a. If the TLV is unknown or the local device has no value for that TLV, it SHALL be skipped and processing will continue with the next TLV. For example, if the device has no Curve25519 Public Point then it would ignore a request for its Curve25519 Public Point TLV.
3. If the TLV ID is equal to the ID of PAN ID Conflict Report Global TLV, then the following SHALL occur.
    a. Construct a PAN ID Conflict Report Global TLV using the current NIB value of nwkPanIdConflictCount.
    b. Set the NIB value nwkPanIdConflictCount to 0.
4. The corresponding TLV SHALL be constructed and appended to the ZDO message.
5. If appending the TLV exceeds the MTU for the message then the following SHALL be done.
    a. Abort processing. Construct and send a Security_Get_Configuration_rsp with a STATUS of FRAME_TOO_LARGE and no other payload.
6. Transmit the Security_Get_Configuration_rsp to the sender of the request.

2.4.3.4.6 Security_Start_Key_Update_req

This command is used by the Trust Center to trigger the receiving device to start its supported link key update mechanism. The Security_Start_Key_Update_req SHALL NOT be APS encrypted or NWK encrypted if the link key update mechanism is done as part of the initial join and before the receiving device has been issued a network key.

The Security_Start_Key_Update_req SHALL be both APS encrypted and NWK encrypted if the link key update mechanism is performed to refresh the link key when the receiving device has the network key and has previously successfully joined the network. The Security_Start_Key_Update_req command (cluster ID = 0x0045) is formatted as illustrated in Figure 2-50.

Octets: Varies
TLVs

Figure 2-50. Format of the Security_Start_Key_Update_req

Table 2-78 specifies the fields of the Security_Start_Key_Update_req command.

Table 2-78. Fields of the Security_Start_Key_Update_req

Name	Type	Valid Range	Description
TLVs	TLV	Varies	The Security_Start_Key_Update_req SHALL include the following TLVs: • Selected Key Negotiation Method TLV • Fragmentation Parameters Global TLV Other TLVs may be included.

2.4.3.4.6.1 Local TLVs

2.4.3.4.6.2 Selected Key Negotiation Method (ID=0)

This indicates the key negotiated method that the sending device would like to negotiate with the receiver. The format is defined in Figure 2-51.

1	**Octets: 1**	**Octets: 8**
Selected Key Negotiation Protocol Enumeration	Selected Pre-shared Secret Enumeration	Sending Device EUI64

Figure 2-51. Selected Key Negotiation Method TLV

Table 2-79 indicates the fields of the Selected Key Negotiation Method TLV.

Table 2-79. Fields of the Selected Key Negotiation Method TLV

Name	Type	Valid Range	Description
Selected Key Negotiation Protocol Enumeration	Enum	0 – 2	The enumeration of the key negotiation method the sender is requesting to use in key negotiation.
Selected Pre-shared Secret Enumeration	Enum	0 – 4	The enumeration indicating the pre-shared secret that the sending device is requesting to be used in the key negotiation.
Sending Device EUI64	EUI64	Any	The value of the EUI64 of the device sending the message. This field SHALL always be present.

Table 2-80 defines the Selected Key Negotiation Protocol Enumeration.

Table 2-80. Selected Key Negotiation Protocol Enumeration

Enumerated Value	Description
0	Reserved (Zigbee 3.0 Mechanism)
1	SPEKE using Curve25519 with Hash AES-MMO-128
2	SPEKE using Curve25519 with Hash SHA-256
3 – 255	Reserved

Table 2-81 defines the Selected Pre-shared Secret Enumeration.

Table 2-81. Selected Pre-shared Secret Enumeration

Enumerated Value	Description
0	Symmetric Authentication Token
1	Pre-configured link-ley derived from installation code
2	Variable-length pass code (for PAKE protocols)
3	Basic Authorization Key
4	Administrative Authorization Key
5 – 254	Reserved
255	Anonymous Well-Known Secret

2.4.3.4.6.3 When Generated

This command is generated by the Trust Center when it wants to trigger the key update process for a device.

2.4.3.4.6.4 Effect on Receipt

On receipt, this command SHALL be processed as follows:

1) If the apsTrustCenterAddress is all F's or if apsTrustCenterAddress is not all F's and the command was not sent by the Trust Center, the receiver SHALL generate a Security_Start_Key_Update_rsp with a status of NOT_AUTHORIZED.
2) If the mandatory TLVs from Table 2-78 are not included, then a Security_Start_Key_Update_rsp SHALL be generated with a status of INV_REQUESTTYPE and no further processing SHALL be done.
3) If apsTrustCenterAddress is unset, the receive SHALL set it with the value of the Sending Device EUI64 field of the Selected Key Negotiation Method TLV.
4) Examine the Selected Key Negotiation Method TLV and determine if the device supports the selected key negotiation methods. If it does not, then a Security_Start_Key_Update_rsp SHALL be generated with a status of NO_MATCH. No further processing SHALL be done.
5) The stack MAY notify the higher layer by passing the contents of the Selected Key Negotiation Method TLV. The stack is responsible for kicking off Key Negotiation or static link key update using one of the locally supported methods.
6) Generate a ZDO Security_Start_Key_Update_rsp with a status of SUCCESS.

2.4.3.4.7 Security_Decommission_req

This command is sent by the Trust Center to inform of the decommissioning of a 3rd party device on the network. The receiving device can use this to clear out any security keys and bindings associated with that 3rd party device. This message SHALL be sent unicast with APS encryption for a centralized network and no APS encryption for a distributed network.

The Security_Decommission_req (Cluster ID=0x0046) is formatted as illustrated in Figure 2-52.

Octets: Varies
TLVs

Figure 2-52. Format of the ZDO Security_Decommission_req Command

Table 2-82 indicates the fields of the ZDO Security_Decommission_req command.

Table 2-82. Fields of the ZDO Security_Decommission_req Command

Name	Type	Valid Range	Description
TLVs	TLVs	Varies	A list of one or more TLVs. The following TLVs have specified behavior in this Revision of the specification: • Device EUI64 List TLV Other TLVs may be included.

2.4.3.4.7.1 Local TLVs

The Local TLVs for the Security_Decommission_req command frame are as follows.

2.4.3.4.7.2 Device EUI64 List TLV (ID=0)

The format of the Device EUI64 List TLV SHALL be as formatted in Figure 2-53.

Octets: 1	8	…
EUI64 Count	EUI64	…

Figure 2-53. Format of the Device EUI64 List TLV

Table 2-83 indicates the fields of the Device EUI64 List TLV.

Table 2-83. Fields of the Device EUI64 List TLV

Name	Type	Valid Range	Description
EUI64 Count	Integer	0x00 – 0xFF	The number of EUI64 fields within the TLV. Note that the maximum value for this count will be dependent on the underlying maximum size of the message as allowed by fragmentation.
EUI64	EUI64	0x0000000000000000 – 0xFFFFFFFFFFFFFFFF	An EUI64 that shall trigger decommissioning operations.

2.4.3.4.7.3 When Generated

This command is generated when the Trust Center has administratively removed a device from the list of authorized devices and wishes to inform other devices about that action. It is NOT used to actually remove that device.

2.4.3.4.7.4 Effect on Receipt

On receipt the following processing SHALL take place.

1) If the command is broadcast it SHALL be silently dropped. No further processing SHALL be done.
2) If the command is unicast on a centralized network with no APS encryption, a ZDO Security_Decommission_rsp SHALL be generated with a status code of NOT_AUTHORIZED. No further processing SHALL be done.
3) If the receiving device is the Trust Center the command SHALL be rejected.
    a) A ZDO Security_Decommission_rsp SHALL be generated with a status code of NOT_AUTHORIZED. No further processing SHALL be done.
4) Execute the General TLV Processing Rules in Annex I.4.7.
5) If the command does not have at least one Device EUI64 List TLV present in the message, it SHALL be rejected.
    a) The receiver SHALL generate a ZDO Security_Decommission_rsp with a status of INV_REQUESTTYPE. No further processing SHALL be done.
6) The receiving device SHALL compare its local EUI64 to all EUI64 in the Security Decommission Req EUI64 TLV. If any EUI64 matches the device's local EUI64 it SHALL be rejected.
    a) The device SHALL generate a ZDO Security_Decommission_rsp with a status of INV_REQUESTTYPE.
7) The receiving device SHALL compare the value of all EUI64 values the Security Decommission Req EUI64 TLV to the DeviceAddress element of all entries in the *apsDeviceKeyPairSet* of the AIB.
    a) If any entry matches it SHALL be deleted.
8) The receiving device SHALL compare the value of all EUI64 in the Security Decommission Req EUI64 TLV to the EUI64 of each binding table entry.
    a) If any entry matches it SHALL be deleted by issuing an APSME-UNBIND.request.
9) Note that the use of the wildcard EUI64 address of 0xFFFFFFFFFFFFFFFF is not allowed and SHALL be ignored.
10) The ZDO MAY inform the NLME of each decommissioned EUI64 allowing the NLME layer to clean up any network layer data related to that device.
11) The device SHALL issue an APS encrypted ZDO Security_Decommission_rsp with the following fields
    a) The Status SHALL be set to SUCCESS if at least one EUI64 matched and resulted in the device making changes to its internal tables.
    b) Otherwise the Status SHALL be set to NOT_FOUND.

2.4.3.4.8 Security_Challenge_req

This command is used by a device to verify the latest frame counter value of another device. The Security_Challenge_req (Cluster ID = 0x0047) is formatted as illustrated in Figure 2-54.

Octets: Varies
TLVs

Figure 2-54. Format of the Security_Challenge_req

2.4.3.4.8.1 Local TLVs

Table 2-84 defines the Local scoped TLVs for this message.

Table 2-84. Global TLVs for Security_Challenge_req

Tag ID	Name
0x00	APS Frame Counter Challenge

2.4.3.4.8.2 APS Frame Counter Challenge TLV

Figure 2-55 illustrates the format of the APS Frame Counter Challenge TLV.

Octets: 8	8
Sender EUI64	Challenge Value

Figure 2-55. Format of the APS Frame Counter Challenge TLV

Table 2-85 describes the fields of the APS Frame Counter Challenge TLV.

Table 2-85. Fields of the APS Frame Counter Challenge TLV

Field	Description
Sender EUI64	The EUI64 of the device that generated the frame.
Challenge Value	A randomly generated 64-bit value sent to a device to prove they have the link key. This allows the initiator to detect replayed challenge response frames.

2.4.3.4.8.3 When Generated

This command is generated when a device wants to challenge another device to verify it has the latest cryptographic data.

This message SHALL NOT be APS encrypted.

2.4.3.4.8.4 Effect on Receipt

1. If the message was broadcast it SHALL be dropped and no further processing SHALL be done.
2. If the message did not include the APS Frame Counter Challenge TLV do the following.
    a. Generate a ZDO Security_Challenge_rsp with a status of MISSING_TLV and send to the device that generated the request.
    b. No further processing SHALL be done.
3. Search the *apsDeviceKeyPairSet* table of the AIB for any entry where the DeviceAddress matches the Sender EUI64 value of the APS Frame Counter Challenge TLV
4. If no match can be found, do the following.
    a. Generate a ZDO Security_Challenge_rsp with a status of NO_MATCH and send to the device that generated the request.
    b. No further processing SHALL be done.
5. Otherwise, follow the procedure in section 4.6.3.8.4.

2.4.4 Server Services

The Device Profile Server Services support the processing of device and service discovery requests, bind requests, unbind requests, and network management requests. Additionally, Server Services support transmission of these responses back to the requesting device.

2.4.4.1 ZDO Response Requirements

A device SHALL be required to support generation of the correct, corresponding ZDO response to all ZDO requests including ZDO messages defined in a future version of this specification. Server Processing marked optional in Table 2-86, Table 2-96, and Table 2-100 allow for the server to use NOT_SUPPORTED as the status code in the response to indicate the lack of support. ZDO requests unknown to the device SHALL be treated as unsupported and also use a NOT_SUPPORTED status code to indicate the device's lack of support for that feature. See below for construction of ZDO responses to unsupported requests. For all broadcast addressed requests (of any broadcast address type) to the server, if the command is not supported, the server SHALL drop the packet. No error status SHALL be unicast back to the Local Device for any broadcast addressed client request including, but not limited to, requests which are not supported on the server.

For all unicast addressed requests to the server, if the command is not supported, the server SHALL formulate a response packet including the response Cluster ID and status fields only. The response Cluster ID SHALL be created by taking the request Cluster ID and setting the high order bit to create the response Cluster ID. The status field SHALL be set to NOT_SUPPORTED. The resulting response SHALL be unicast to the requesting client.

2.4.4.2 Device and Service Discovery Server

Table 2-86 lists the commands supported by the Device and Service Discovery Server Services device profile. Each of these commands will be discussed in the following sections. For receipt of the Device_annce command, the server SHALL check all internal references to the IEEE and 16-bit NWK addresses supplied in the request. For all references to the IEEE address in the Local Device, the corresponding NWK address supplied in the Device_annce SHALL be substituted. For any other references to the NWK address in the Local Device, the corresponding entry SHALL be marked as not having a known valid 16-bit NWK address, even if the IEEEAddr field in the message carries the value of 0xffffffffffffffff. The server SHALL NOT supply a response to the Device_annce.

Table 2-86. Device and Service Discovery Server Service Primitives

Device and Service Discovery Server Services	Cluster ID	Server Processing
NWK_addr_rsp	0x8000	M
IEEE_addr_rsp	0x8001	M
Node_Desc_rsp	0x8002	M
Power_Desc_rsp	0x8003	M
Simple_Desc_rsp	0x8004	M
Active_EP_rsp	0x8005	M
Match_Desc_rsp	0x8006	M
Complex_Desc_rsp	0x8010	Deprecated

Device and Service Discovery Server Services	Cluster ID	Server Processing
User_Desc_rsp	0x8011	Deprecated
User_Desc_conf	0x8014	Deprecated
Parent_annce_rsp	0x801f	M
System_Server_Discovery_rsp	0x8015	O
Discovery_store_rsp	0x8016	Deprecated
Node_Desc_store_rsp	0x8017	Deprecated
Power_Desc_store_rsp	0x8018	Deprecated
Active_EP_store_rsp	0x8019	Deprecated
Simple_Desc_store_rsp	0x801a	Deprecated
Remove_node_cache_rsp	0x801b	Deprecated
Find_node_cache_rsp	0x801c	Deprecated
Extended_Simple_Desc_rsp	0x801d	Deprecated
Extended_Active_EP_rsp	0x801e	Deprecated

2.4.4.2.1 NWK_addr_rsp

The NWK_addr_rsp command (ClusterID=0x8000) SHALL be formatted as illustrated in Figure 2-56.

Octets: 1	8	2	0/1	0/1	Variable
Status	IEEEAddr RemoteDev	NWKAddr RemoteDev	Num AssocDev	StartIndex	NWKAddr AssocDevList

Figure 2-56. Format of the NWK_addr_rsp Command Frame

Table 2-87 specifies the fields of the NWK_addr_rsp command frame.

Table 2-87. Fields of the NWK_addr_rsp Command Frame

Name	Type	Valid Range	Description
Status	Integer	SUCCESS, INV_REQUESTTYPE, or DEVICE_NOT_FOUND	The status of the NWK_addr_req command.

Name	**Type**	**Valid Range**	**Description**
IEEEAddrRemoteDev	Device Address	An extended 64-bit, IEEE address	64-bit address for the Remote Device.
NWKAddrRemoteDev	Device Address	A 16-bit, NWK address	16-bit address for the Remote Device.
NumAssocDev	Integer	0x00 – 0xff	Count of the number of 16-bit short addresses to follow. If the RequestType in the request is Extended Response and there are no associated devices on the Remote Device, this field SHALL be set to 0. If an error occurs or the Request Type in the request is for a Single Device Response, this field SHALL NOT be included in the frame.
StartIndex	Integer	0x00 – 0xff	Starting index into the list of associated devices for this report. If the RequestType in the request is Extended Response and there are no associated devices on the Remote Device, this field SHALL NOT be included in the frame. If an error occurs or the Request Type in the request is for a Single Device Response, this field SHALL NOT be included in the frame.
NWKAddrAssocDevList	Device Address List	List of NumAssocDev 16-bit short addresses, each with range 0x0000 – 0xffff	A list of 16-bit addresses, one corresponding to each associated device to Remote Device; The number of 16-bit network addresses contained in this field is specified in the NumAssocDev field. If the RequestType in the request is Extended Response and there are no associated devices on the Remote Device, this field SHALL NOT be included in the frame. If an error occurs or the Request Type in the request is for a Single Device Response, this field SHALL NOT be included in the frame.

2.4.4.2.1.1 When Generated

The NWK_addr_rsp is generated by a Remote Device in response to a NWK_addr_req command inquiring as to the NWK address of the Remote Device or the NWK address of an address held in the neighbor table (see section 2.4.3.1.1.2 for a detailed description). The destination addressing on this command is unicast.

2.4.4.2.1.2 Effect on Receipt

On receipt of the NWK_addr_rsp command, the recipient is either notified of the status of its attempt to discover a NWK address from an IEEE address or notified of an error. If the NWK_addr_rsp command is received with a Status of SUCCESS, the remaining fields of the command contain the appropriate discovery information, according to the RequestType as specified in the original NWK_Addr_req command. Otherwise, the Status field indicates the error and the NumAssocDev, StartIndex, and NWKAddrAssocDevList fields SHALL NOT be included.

2.4.4.2.2 IEEE_addr_rsp

The IEEE_addr_rsp command (ClusterID=0x8001) SHALL be formatted as illustrated in Figure 2-57.

Octets: 1	8	2	0/1	0/1	Variable
Status	IEEEAddr RemoteDev	NWKAddr RemoteDev	NumAssocDev	StartIndex	NWKAddr AssocDevList

Figure 2-57. Format of the IEEE_addr_rsp Command Frame

Table 2-88 specifies the fields of the IEEE_addr_rs command frame.

Table 2-88. Fields of the IEEE_addr_rsp Command Frame

Name	Type	Valid Range	Description
Status	Integer	SUCCESS, INV_REQUESTTYPE or DEVICE_NOT_FOUND	The status of the IEEE_addr_req command.
IEEEAddrRemoteDev	Device Address	An extended 64-bit, IEEE address	64-bit address for the Remote Device.
NWKAddrRemoteDev	Device Address	A 16-bit, NWK address	16-bit address for the Remote Device.
NumAssocDev	Integer	0x00 – 0xff	Count of the number of 16-bit short addresses to follow. If the RequestType in the request is Extended Response and there are no associated devices on the Remote Device, this field SHALL be set to 0. If an error occurs or the RequestType in the request is for a Single Device Response, this field SHALL NOT be included in the frame.

Name	Type	Valid Range	Description
StartIndex	Integer	0x00 – 0xff	Starting index into the list of associated devices for this report. If the RequestType in the request is Extended Response and there are no associated devices on the Remote Device, this field SHALL NOT be included in the frame. If an error occurs or the RequestType in the request is for a Single Device Response, this field SHALL NOT be included in the frame.
NWKAddrAssocDevList	Device Address List	List of NumAssocDev 16-bit short addresses, each with range 0x0000 – 0xffff	A list of 16-bit addresses, one corresponding to each associated device to Remote Device; The number of 16-bit network addresses contained in this field is specified in the NumAssocDev field. If the RequestType in the request is Extended Response and there are no associated devices on the Remote Device, this field SHALL NOT be included in the frame. If an error occurs or the RequestType in the request is for a Single Device Response, this field SHALL NOT be included in the frame

2.4.4.2.2.1 When Generated

The IEEE_addr_rsp is generated by a Remote Device in response to an IEEE_addr_req command inquiring as to the 64-bit IEEE address of the Remote Device or the 64-bit IEEE address of an address held in the neighbor table (see section 2.4.3.1.2.2 for a detailed description). The destination addressing on this command SHALL be unicast.

2.4.4.2.2.2 Effect on Receipt

On receipt of the IEEE_addr_rsp command, the recipient is either notified of the status of its attempt to discover an IEEE address from an NWK address or notified of an error. If the IEEE_addr_rsp command is received with a Status of SUCCESS, the remaining fields of the command contain the appropriate discovery information, according to the RequestType as specified in the original IEEE_Addr_req command. Otherwise, the Status field indicates the error and the NumAssocDev, StartIndex, and NWKAddrAssocDevList fields SHALL NOT be included.

2.4.4.2.3 Node_Desc_rsp

The Node_Desc_rsp command (ClusterID=0x8002) SHALL be formatted as illustrated in Figure 2-58.

Octets: 1	2	See section 2.3.2.3	TLVs
Status	NWKAddrOfInterest	Node Descriptor	One or more TLVs

Figure 2-58. Format of the Node_Desc_rsp Command Frame

Table 2-89 specifies the fields of the Node_Desc_rsp command frame.

Table 2-89. Fields of the Node_Desc_rsp Command Frame

Name	Type	Valid Range	Description
Status	Integer	SUCCESS, DEVICE_NOT_FOUND, INV_REQUESTTYPE, or NO_DESCRIPTOR	The status of the Node_Desc_req command.
NWKAddrOfInterest	Device Address	16-bit NWK address	NWK address for the request.
NodeDescriptor	Node Descriptor		See the Node Descriptor format in section 2.3.2.3. This field SHALL only be included in the frame if the status field is equal to SUCCESS.
TLVs	TLV List	Varies	A set of TLVs. The Fragmentation Parameters Global TLV SHALL always be included.

2.4.4.2.3.1 Local TLVs

2.4.4.2.3.2 Selected Key Negotiation Method (ID=0)

This TLV has the same format and ID as the Selected Key Negotiation Method TLV of the Security_Start_Key_Update_req.

2.4.4.2.3.3 When Generated

The Node_Desc_rsp is generated by a remote device in response to a Node_Desc_req directed to the remote device. This command SHALL be unicast to the originator of the Node_Desc_req command.

If the Node_Desc_req frame includes the Fragmentation Parameters Global TLV the receiver can cache the information in the *apsFragmentationCacheTable*. See section 2.4.4.2.3.4 for more information.

If the Node_Desc_req frame includes at least one valid TLV, the receiver SHALL set the Frame Counter Synchronization bit in the Features & Capabilities bitmap of the apsDeviceKeyPairSet entry pertaining to the sender of the Node_Desc_req command to '1', if such an entry exists.

The remote device SHALL generate the Node_Desc_rsp command using the format illustrated in Figure 2-58. The NWKAddrOfInterest field SHALL match that specified in the original Node_Desc_req command. If the NWKAddrOfInterest field matches the network address of the remote device, it SHALL set the Status field to SUCCESS and include its node descriptor (see section 2.3.2.3) in the NodeDescriptor field.

If the NWKAddrOfInterest field does not match the network address of the remote device and it is an end device, it SHALL set the Status field to INV_REQUESTTYPE, set the ActiveEPCount field to 0, and not include the ActiveEPList field. If the NWKAddrOfInterest field does not match the network address of the remote device and it is the coordinator or a router, it SHALL set the Status field to DEVICE_NOT_FOUND, set the ActiveEPCount field to 0, and not include the ActiveEPList field. If the NWKAddrOfInterest matches the network address of one of the children of the remote device, it SHALL determine whether a node descriptor for that device is available. If a node descriptor is not available for the child indicated by the NWKAddrOfInterest field, the remote device SHALL set the Status field to NO_DESCRIPTOR and not include the NodeDescriptor field. If a node descriptor is available for the child indicated by the NWKAddrOfInterest field, the remote device SHALL set the Status field to SUCCESS and include the node descriptor (see section 2.3.2.3) of the matching child device in the NodeDescriptor field.

The device sending the Node_Desc_rsp SHALL include the following TLVs:

- Selected Key Negotiation Method TLV.
- Fragmentation Parameters Global TLV

Devices prior to Revision 23 will not include the TLV field. The receiver SHALL still accept messages without TLVs in the response message.

2.4.4.2.3.4 Effect on Receipt

On receipt of the Node_Desc_rsp command, the recipient is either notified of the node descriptor of the remote device indicated in the original Node_Desc_req command or notified of an error. If the Node_Desc_rsp command is received with a Status of SUCCESS, the NodeDescriptor field SHALL contain the requested node descriptor. Otherwise, the Status field indicates the error and the NodeDescriptor field SHALL NOT be included.

The receiver can use the Fragmentation Parameters Global TLV to cache the sender's fragmentation capabilities in the *apsFragmentationCacheTable*. The Trust Center SHALL cache the data for all devices in the network. A regular device SHALL cache fragmentation support for the Trust Center and MAY cache data for any other device in the network.

If the core stack Revision indicated in the Node_Desc_rsp is 23 or higher, the receiver SHALL set the Frame Counter Synchronization bit in the Features & Capabilities bitmap of the apsDeviceKeyPairSet entry pertaining to the sender of the Node_Desc_rsp command to '1', if such an entry exists.

2.4.4.2.4 Power_Desc_rsp

The Power_Desc_rsp command (ClusterID=0x8003) SHALL be formatted as illustrated in Figure 2-59.

Octet: 1	**2**	**Variable**
Status	NWKAddrOfInterest	Power Descriptor

Figure 2-59. Format of the Power_Desc_rsp Command Frame

Table 2-90 specifies the fields of the Power_Desc_rsp command frame.

Table 2-90. Fields of the Power_Desc_rsp Command Frame

Name	**Type**	**Valid Range**	**Description**
Status	Integer	SUCCESS, DEVICE_NOT_FOUND, INV_REQUESTTYPE, or NO_DESCRIPTOR	The status of the Power_Desc_req command.
NWKAddrOfInterest	Device Address	16-bit NWK address	NWK address for the request.

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

2.4.4.2.4.1 When Generated

The Power_Desc_rsp is generated by a remote device in response to a Power_Desc_req directed to the remote device. This command SHALL be unicast to the originator of the Power_Desc_req command.

The remote device SHALL generate the Power_Desc_rsp command using the format illustrated in . The NWKAddrOfInterest field SHALL match that specified in the original Power_Desc_req command. If the NWKAddrOfInterest field matches the network address of the remote device, it SHALL set the Status field to SUCCESS and include its power descriptor (see section 2.3.2.4) in the PowerDescriptor field.

If the NWKAddrOfInterest field does not match the network address of the remote device and it is an end device, it SHALL set the Status field to INV_REQUESTTYPE and not include the PowerDescriptor field. If the NWKAddrOfInterest field does not match the network address of the remote device and it is the coordinator or a router, it SHALL determine whether the NWKAddrOfInterest field matches the network address of one of its children. If the NWKAddrOfInterest field does not match the network address of one of the children of the remote device, it SHALL set the Status field to DEVICE_NOT_FOUND and not include the PowerDescriptor field. If the NWKAddrOfInterest matches the network address of one of the children of the remote device, it SHALL determine whether a power descriptor for that device is available. If a power descriptor is not available for the child indicated by the NWKAddrOfInterest field, the remote device SHALL set the Status field to NO_DESCRIPTOR and not include the PowerDescriptor field. If a power descriptor is available for the child indicated by the NWKAddrOfInterest field, the remote device SHALL set the Status field to SUCCESS and include the power descriptor (see section 2.3.2.4) of the matching child device in the PowerDescriptor field.

2.4.4.2.4.2 Effect on Receipt

On receipt of the Power_Desc_rsp command, the recipient is either notified of the power descriptor of the remote device indicated in the original Power_Desc_req command or notified of an error. If the Power_Desc_rsp command is received with a Status of SUCCESS, the PowerDescriptor field SHALL contain the requested power descriptor. Otherwise, the Status field indicates the error and the PowerDescriptor field SHALL NOT be included.

2.4.4.2.5 Simple_Desc_rsp

The Simple_Desc_rsp command (ClusterID=0x8004) SHALL be formatted as illustrated in Figure 2-60.

Octet: 1	2	1	Variable
Status	NWKAddrOfInterest	Length	Simple Descriptor

Figure 2-60. Format of the Simple_Desc_rsp Command Frame

Table 2-91 specifies the fields of the Simple_Desc_rsp command frame.

Table 2-91. Fields of the Simple_Desc_rsp Command Frame

Name	Type	Valid Range	Description
Status	Integer	SUCCESS, INVALID_EP, NOT_ACTIVE, DEVICE_NOT_FOUND, INV_REQUESTTYPE or NO_DESCRIPTOR	The status of the Simple_Desc_req command.
NWKAddrOfInterest	Device Address	16-bit NWK address	NWK address for the request.
Length	Integer	0x00 – 0xff	Length in bytes of the Simple Descriptor to follow.
SimpleDescriptor	Simple Descriptor		See the Simple Descriptor format in section 2.3.2.5. This field SHALL only be included in the frame if the status field is equal to SUCCESS.

2.4.4.2.5.1 When Generated

The Simple_Desc_rsp is generated by a remote device in response to a Simple_Desc_req directed to the remote device. This command SHALL be unicast to the originator of the Simple_Desc_req command.

The remote device SHALL generate the Simple_Desc_rsp command using the format illustrated in . The NWKAddrOfInterest field SHALL match that specified in the original Simple_Desc_req command. If the endpoint field specified in the original Simple_Desc_req command does not fall within the correct range specified in Table 2-91, the remote device SHALL set the Status field to INVALID_EP, set the Length field to 0 and not include the SimpleDescriptor field.

If the NWKAddrOfInterest field matches the network address of the remote device, it SHALL determine whether the endpoint field specifies the identifier of an active endpoint on the device. If the endpoint field corresponds to an active endpoint, the remote device SHALL set the Status field to SUCCESS, set the Length field to the length of the simple descriptor on that endpoint, and include the simple descriptor (see section 2.3.2.5) for that endpoint in the SimpleDescriptor field. If the endpoint field does not correspond to an active endpoint, the remote device SHALL set the Status field to NOT_ACTIVE, set the Length field to 0, and not include the SimpleDescriptor field.

If the NWKAddrOfInterest field does not match the network address of the remote device and it is an end device, it SHALL set the Status field to INV_REQUESTTYPE, set the Length field to 0, and not include the SimpleDescriptor field. If the NWKAddrOfInterest field does not match the network address of the remote device and it is the coordinator or a router, it SHALL determine whether the NWKAddrOfInterest field matches the network address of one of its children. If the NWKAddrOfInterest field does not match the network address of one of the children of the remote device, it SHALL set the Status field to DEVICE_NOT_FOUND, set the Length field to 0, and not include the SimpleDescriptor field.

If the NWKAddrOfInterest matches the network address of one of the children of the remote device, it SHALL determine whether a simple descriptor for that device and on the requested endpoint is available. If a simple descriptor is not available on the requested endpoint of the child indicated by the NWKAddrOfInterest field, the remote device SHALL set the Status field to NO_DESCRIPTOR, set the Length field to 0, and not include the SimpleDescriptor field. If a simple descriptor is available on the requested endpoint of the child indicated by the NWKAddrOfInterest field, the remote device SHALL set the Status field to SUCCESS, set the Length field to the length of the simple descriptor on that endpoint, and include the simple descriptor (see section 2.3.2.5) for that endpoint of the matching child device in the SimpleDescriptor field.

2.4.4.2.5.2 Effect on Receipt

On receipt of the Simple_Desc_rsp command, the recipient is either notified of the simple descriptor on the endpoint of the remote device indicated in the original Simple_Desc_req command or notified of an error. If the Simple_Desc_rsp command is received with a Status of SUCCESS, the SimpleDescriptor field SHALL contain the requested simple descriptor. Otherwise, the Status field indicates the error and the SimpleDescriptor field SHALL NOT be included.

2.4.4.2.6 Active_EP_rsp

The Active_EP_rsp command (ClusterID=0x8005) SHALL be formatted as illustrated in Figure 2-61.

Octet: 1	**2**	**1**	**Variable**
Status	NWKAddrOfInterest	ActiveEPCount	ActiveEPList

Figure 2-61. Format of the Active_EP_rsp Command Frame

Table 2-92 specifies the fields of the Active_EP_rsp command frame.

Table 2-92. Fields of the Active_EP_rsp Command Frame

Name	Type	Valid Range	Description
Status	Integer	SUCCESS, DEVICE_NOT_FOUND, INV_REQUESTTYPE, or NO_DESCRIPTOR	The status of the Active_EP_req command.
NWKAddrOfInterest	Device Address	16-bit NWK address	NWK address for the request.
ActiveEPCount	Integer	0x00 – 0xff	The count of active endpoints on the Remote Device.
ActiveEPList			List of bytes each of which represents an 8-bit endpoint.

2.4.4.2.6.1 When Generated

The Active_EP_rsp is generated by a remote device in response to an Active_EP_req directed to the remote device. This command SHALL be unicast to the originator of the Active_EP_req command.

The remote device SHALL generate the Active_EP_rsp command using the format illustrated in . The NWKAddrOfInterest field SHALL match that specified in the original Active_EP_req command. If the NWKAddrOfInterest field matches the network address of the remote device, it SHALL set the Status field to SUCCESS, set the ActiveEPCount field to the number of active endpoints on that device and include an ascending list of all the identifiers of the active endpoints on that device in the ActiveEPList field.

If the NWKAddrOfInterest field does not match the network address of the remote device and it is an end device, it SHALL set the Status field to INV_REQUESTTYPE, set the ActiveEPCount field to 0, and not include the ActiveEPList field. If the NWKAddrOfInterest field does not match the network address of the remote device and it is the coordinator or a router, it SHALL determine whether the NWKAddrOfInterest field matches the network address of a device it holds in a discovery cache. If the NWKAddrOfInterest field does not match the network address of a device it holds in a discovery cache, it SHALL set the Status field to DEVICE_NOT_FOUND, set the ActiveEPCount field to 0, and not include the ActiveEPList field. If the NWKAddrOfInterest matches the network address of a device held in a discovery cache on the remote device, it SHALL determine whether that device has any active endpoints. If the discovery information corresponding to the ActiveEP request has not yet been uploaded to the discovery cache, the remote device SHALL set the Status field to NO_DESCRIPTOR, set the ActiveEPCount field to 0 and not include the ActiveEPList field. If the cached device has no active endpoints, the remote device SHALL set the Status field to SUCCESS, set the ActiveEPCount field to 0, and not include the ActiveEPList field. If the cached device has active endpoints, the remote device SHALL set the Status field to SUCCESS, set the ActiveEPCount field to the number of active endpoints on that device, and include an ascending list of all the identifiers of the active endpoints on that device in the ActiveEPList field.

2.4.4.2.6.2 Effect on Receipt

On receipt of the Active_EP_rsp command, the recipient is either notified of the active endpoints of the remote device indicated in the original Active_EP_req command or notified of an error. If the Active_EP_rsp command is received with a Status of SUCCESS, the ActiveEPCount field indicates the number of entries in the ActiveEPList field. Otherwise, the Status field indicates the error and the ActiveEPList field SHALL NOT be included.

2.4.4.2.7 Match_Desc_rsp

The Match_Desc_rsp command (ClusterID=0x8006) SHALL be formatted as illustrated inFigure 2-62.

Octet: 1	2	1	Variable
Status	NWKAddrOfInterest	Match Length	Match List

Figure 2-62. Format of the Match_Desc_rsp Command Frame

Table 2-93 specifies the fields of the Match_Desc_rsp command frame.

Table 2-93. Fields of the Match_Desc_rsp Command Frame

Name	Type	Valid Range	Description
Status	Integer	SUCCESS, DEVICE_NOT_FOUND, INV_REQUESTTYPE, or NO_DESCRIPTOR	The status of the Match_Desc_req command.
NWKAddrOfInterest	Device Address	16-bit NWK address	NWK address for the request.
MatchLength	Integer	0x00-0xff	The count of endpoints on the Remote Device that match the request criteria.
MatchList			List of bytes each of which represents an 8-bit endpoint.

2.4.4.2.7.1 When Generated

The Match_Desc_rsp is generated by a remote device in response to a Match_Desc_req either broadcast or unicast to the remote device. This command SHALL be unicast to the originator of the Match_Desc_req command.

The following describes the procedure for processing the Match_Desc_req and generation of Match_Desc_rsp.

1. Set MatchLength to 0 and create an empty list MatchList.
2. If the receiving device is an End Device and the NWKAddrOfInterest within the Match_Desc_req message does not match the nwkNetworkAddress of the NIB and is not a broadcast address, the following SHALL be performed. Otherwise it shall proceed to step 3.
    a. If the NWK destination of the message is a broadcast address, no further processing SHALL be done.
    b. If the NWK destination is a unicast address, the following SHALL be performed.
        i. Set the Status value to INV_REQUESTTYPE.
        ii. Set the MatchLength to 0.
        iii. Construct a Match_Desc_rsp with only Status and MatchLength fields.
        iv. Send the message as a unicast to the source of the Match_Desc_req.
        v. No further processing SHALL be done.
3. If the NWKAddrOfInterest is equal to the nwkNetworkAddress of the NIB, or is a broadcast address, perform the following procedure. Otherwise proceed to step 4.
    a. Apply the match criteria in section 2.4.4.2.7.2 for all local Simple Descriptors.
    b. For each Simple Descriptor that matches with at least one cluster, add the endpoint once to MatchList and increment MatchLength.
4. If the NWKAddrOfInterest is not a broadcast address, the NWKAddressOfInterest is not equal to the nwkNetworkAddress of the local NIB, and the device is a coordinator or router, then the following SHALL be performed. Otherwise proceed to step 5.
    a. Examine each entry in the nwkNeighborTable and perform the following procedure.
        i. If the Network Address of the entry does not match the NWKAddrOfInterest or the Device Type is not equal to 0x02 (Zigbee End Device), do not process this entry. Continue to the next entry in the nwkNeighborTable.
        ii. For each endpoint that matches with at least once cluster, add that endpoint once to the MatchList and increment MatchLength.
        iii. Proceed to step 7.
    b. If the NWKAddrOfInterest does not match any entry in the nwkNeighborTable, perform the following:
        i. Set the Status to DEVICE_NOT_FOUND.
        ii. Construct a Match_Desc_rsp with Status and MatchLength fields only.
        iii. Unicast the message to the source of the Match_Desc_req.
        iv. No further processing SHALL be done.
5. If the MatchLength is 0 and the NWK destination of the Match_Desc_req was a broadcast address, no further processing SHALL be done. Otherwise proceed to step 6.
6. If the MatchLength is 0 and the NWKAddrOfInterest matched an entry in the nwkNeighborTable, the following SHALL be performed. Otherwise proceed to step 7.
    a. Set the Status to NO_DESCRIPTOR
    b. Construct a Match_Desc_rsp with Status and MatchLength only.
    c. Unicast the Match_Desc_rsp to the source of the Match_Desc_req.
    d. No further processing SHALL be done.
7. The following SHALL be performed. This is the case for both MatchLength > 0 and MatchLength == 0.
    a. Set the Status to SUCCESS.

b. Construct a Match_Desc_rsp with Status, NWKAddrOfInterest, MatchLength, and MatchList.

c. Unicast the response to the NWK source of the Match_Desc_req.

2.4.4.2.7.2 Simple Descriptor Matching Rules

These rules will examine a ProfileID, InputClusterList, OutputClusterList, and a SimpleDescriptor. The following SHALL be performed:

1. The device SHALL first check if the ProfileID field matches using the Profile ID of the SimpleDescriptor. If the profile identifiers do not match and the ProfileID is not 0xffff, the device SHALL note the match as unsuccessful and no further processing SHALL be done.
2. Examine the InputClusterList and compare each item to the Application Input Cluster List of the SimpleDescriptor.
    a. If a cluster ID matches exactly, then the device SHALL note the match as successful and perform no further matching. Processing is complete.
3. Examine the OutputClusterList and compare each item to the Application Output Cluster List of the SimpleDescriptor.
    a. If a cluster ID matches exactly, then the device SHALL note the match as successful and perform no further matching. Processing is complete.
4. The device SHALL note the match as unsuccessful. Processing is complete.

2.4.4.2.7.3 Effect on Receipt

On receipt of the Match_Desc_rsp command, the recipient is either notified of the results of its match criterion query indicated in the original Match_Desc_req command or notified of an error. If the Match_Desc_rsp command is received with a Status of SUCCESS, the MatchList field SHALL contain the list of endpoints containing simple descriptors that matched the criterion. Otherwise, the Status field indicates the error and the MatchList field SHALL NOT be included.

2.4.4.2.8 Complex_Desc_rsp – DEPRECATED

2.4.4.2.9 User_Desc_rsp – DEPRECATED

2.4.4.2.10 System_Server_Discovery_rsp

The System_Server_Discovery_rsp command (ClusterID=0x8015) SHALL be formatted as illustrated in Figure 2-63.

Octet: 1	2
Status	ServerMask

Figure 2-63. System_Server_Discovery_rsp Command Frame

Table 2-94 specifies the fields of the System_Server_Discovery_rsp command frame.

Table 2-94. Fields of the System_Server_Discovery_rsp Command Frame

Name	Type	Valid Range	Description
Status	Integer	SUCCESS	The status of the System_Server_Discovery_rsp command.
ServerMask	Integer	Bitmap	See Table 2-34 for bit assignments.

2.4.4.2.10.1 When Generated

The System_Server_Discovery_rsp is generated from Remote Devices on receipt of a System_Server_Discovery_req primitive if the parameter matches the Server Mask field in its node descriptor. If there is no match, the System_Server_Discovery_req SHALL be ignored and no response given. Matching is performed by masking the ServerMask parameter of the System_Server_Discovery_req with the Server Mask field in the node descriptor. This command SHALL be unicast to the device which sent System_Server_Discovery_req with Acknowledge request set in TxOptions. The parameter ServerMask contains the bits in the parameter of the request which match the server mask in the node descriptor.

2.4.4.2.10.2 Effect on Receipt

The requesting device is notified that this device has some of the system server functionality that the requesting device is seeking.

If the Network Manager bit was set in the System_Server_Discovery_rsp, then the Remote Device's NWK address SHALL be set into the *nwkManagerAddr* of the NIB.

2.4.4.2.11 User_Desc_conf – DEPRECATED

2.4.4.2.12 Discovery_Cache_rsp – DEPRECATED

2.4.4.2.13 Discovery_store_rsp – DEPRECATED

2.4.4.2.14 Node_Desc_store_rsp – DEPRECATED

2.4.4.2.15 Power_Desc_store_rsp – DEPRECATED

2.4.4.2.16 Active_EP_store_rsp– DEPRECATED

2.4.4.2.17 Simple_Desc_store_rsp – DEPRECATED

2.4.4.2.18 Find_node_cache_rsp – DEPRECATED

2.4.4.2.19 Extended_Simple_Desc_rsp – DEPRECATED

2.4.4.2.20 Extended_Active_EP_rsp – DEPRECATED

2.4.4.2.21 Remove_node_cache_rsp – DEPRECATED

2.4.4.2.22 Parent_annce_rsp

The Parent_annce_rsp command (ClusterID = 0x801f) SHALL be formatted as illustrated in Figure 2-64, and is generated in response to a Parent_annce.

Octets: 1	1	Variable	…	Variable
Status	NumberOfChildren	ChildInfo[0]	…	ChildInfo[n]

Figure 2-64. Format of the Parent_annce_rsp Command Frame

Table 2-95 specifies the fields of the Parent_annce_rsp command frame.

Table 2-95. Fields of the Parent_annce_rsp Command Frame

Name	Type	Valid Range	Description
Status	Integer	SUCCESS or NOT_SUPPORTED	The status of the Parent_annce command.
NumberOfChildren	Integer	0 – 255	The number of ChildInfo structures contained in the message.
ChildInfo	ChildInfo	Variable	The child information. See Table 2-52.

Table 2-52 specifies the contents of the ChildInfo structure. This is the same format as the Parent_annce.

2.4.4.2.22.1 When Generated

Upon receipt of a Parent_annce message, a router SHALL construct but not yet send a Parent_annce_rsp message with the NumberOfChildren field set to 0. It SHALL then examine each Extended Address present in the Parent_annce message and search its Neighbor Table for an entry that matches. If a device is found and the Device Type is Zigbee end device (0x02), the router SHALL do the following.

1. If the Keepalive Received value is TRUE, it SHALL keep the parent/child relationship in the neighbor table unmodified. It SHALL then do the following:
    a. Append the ChildInfo structure to the Parent_annce_rsp.
    b. Increment NumberOfChildren by 1.
2. If the Keepalive Received value is FALSE, it SHALL remove the entry.

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

If the NumberOfChildren field in the Parent_Annce_rsp is greater than 0, it SHALL unicast the message to the sender of the Parent_Annce message.

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

2.4.4.2.22.2 Effect on Receipt

On receipt of a Parent_annce_rsp, the device SHALL examine its Neighbor Table for each extended address in the ChildInfo entry and do the following.

i) If the entry matches and the Device Type is Zigbee End Device (0x02), it SHALL do the following:
    (1) Delete the entry from the Neighbor table.

ii) If the entry does not match, no more processing is performed on this ChildInfo entry.

There is no message generated in response to a Parent_annce_rsp.Bind, Unbind Bind Management Server Services.

Table 2-96 lists the commands supported by Device Profile: Bind and Unbind Server Services. Each of these primitives will be discussed in the following sections.

Table 2-96. Unbind and Bind Management Server Services Primitives

Bind and Unbind Server Service Commands	Cluster ID	Server Processing
End_Device_Bind_rsp	0x8020	Deprecated
Bind_rsp	0x8021	O
Unbind_rsp	0x8022	O
Bind_Register_rsp	0x8023	Deprecated
Replace_Device_rsp	0x8024	Deprecated
Store_Bkup_Bind_Entry_rsp	0x8025	Deprecated
Remove_Bkup_Bind_Entry_rsp	0x8026	Deprecated
Backup_Bind_Table_rsp	0x8027	Deprecated
Recover_Bind_Table_rsp	0x8028	Deprecated
Backup_Source_Bind_rsp	0x8029	Deprecated
Recover_Source_Bind_rsp	0x802a	Deprecated
Clear_All_Bindings_rsp	0x802b	O

2.4.4.2.23 End_Device_Bind_rsp – DEPRECATED

2.4.4.2.24 Bind_rsp

The Bind_rsp command (ClusterID=0x8021) SHALL be formatted as illustrated in Figure 2-65.

Octets: 1
Status

Figure 2-65. Format of the Bind_rsp Command Frame

Table 2-97 specifies the fields of the Bind_rsp command frame.

Table 2-97. Fields of the Bind_rsp Command Frame

Name	Type	Valid Range	Description
Status	Integer	SUCCESS, NOT_SUPPORTED, INVALID_EP, TABLE_FULL, or NOT_AUTHORIZED	The status of the Bind_req command.

2.4.4.2.24.1 When Generated

The Bind_rsp is generated in response to a Bind_req. If the Bind_req is processed and the Binding Table entry committed on the Remote Device, a Status of SUCCESS is returned. If the Remote Device is not a Primary binding table cache or the SrcAddress, a Status of NOT_SUPPORTED is returned. The endpoint of the Bind_req SHALL be checked to determine whether it is between the inclusive range of 0x01 to 0xFE, and if not a Bind_rsp SHALL be generated with a status of INVALID_EP.

2.4.4.2.24.2 Effect on Receipt

Upon receipt, error checking is performed on the request as described in the previous section. Assuming the Status is SUCCESS, the parameters from the Bind_req are entered into the Binding Table at the Remote Device via the APSME-BIND.request primitive.

2.4.4.2.25 Unbind_rsp

The Unbind_rsp command (ClusterID=0x8022) SHALL be formatted as illustrated in Figure 2-66.

Octets: 1
Status

Figure 2-66. Format of the Unbind_rsp Command Frame

Table 2-98 specifies the fields of the Unbind_rsp command frame.

Table 2-98. Fields of the Unbind_rsp Command Frame

Name	Type	Valid Range	Description
Status	Integer	SUCCESS, NOT_SUPPORTED, INVALID_EP, NO_ENTRY, or NOT_AUTHORIZED	The status of the Unbind_req command.

2.4.4.2.25.1 When Generated

The Unbind_rsp is generated in response to an Unbind_req. If the Unbind_req is processed and the corresponding Binding Table entry is removed from the Remote Device, a Status of SUCCESS is returned. If the Remote Device is not the Zigbee Coordinator or the SrcAddress, a Status of NOT_SUPPORTED is returned. The supplied endpoint SHALL be checked to determine whether it falls within the specified range. If it does not, a Status of INVALID_EP SHALL be returned. If the Remote Device is the Zigbee Coordinator or SrcAddress but does not have a Binding Table entry corresponding to the parameters received in the request, a Status of NO_ENTRY is returned.

2.4.4.2.25.2 Effect on Receipt

Upon receipt, error checking is performed on the response. If the status is SUCCESS, the device has successfully removed the binding entry for the parameters specified in the Unbind_req.

2.4.4.2.26 Bind_Register_rsp – DEPRECATED

2.4.4.2.27 Replace_Device_rsp – DEPRECATED

2.4.4.2.28 Store_Bkup_Bind_Entry_rsp – DEPRECATED

2.4.4.2.29 Remove_Bkup_Bind_Entry_rsp – DEPRECATED

2.4.4.2.30 Backup_Bind_Table_rsp – DEPRECATED

2.4.4.2.31 Recover_Bind_Table_rsp – DEPRECATED

2.4.4.2.32 Backup_Source_Bind_rsp – DEPRECATED

2.4.4.2.33 Recover_Source_Bind_rsp – DEPRECATED

2.4.4.2.34 Clear_All_Bindings_rsp

The Clear_All_Binding_rsp command (ClusterID=0x802b) SHALL be formatted as illustrated in Figure 2-67.

Octets: 1
Status

Figure 2-67. Format of the Clear_All_Bindings_rsp Command Frame

Table 2-99 specifies the fields of the Unbind_rsp command frame.

Table 2-99. Fields of the Clear_All_Bindings_rsp Command Frame

Name	Type	Valid Range	Description
Status	Integer	SUCCESS, NOT_SUPPORTED, NOT_AUTHORIZED, INV_REQUESTTYPE, or NO_MATCH.	The status of the ZDO Clear_All_Bindings_req.

2.4.4.2.34.1 When Generated

This command is generated in response to a ZDO Clear_All_Bindings_req.

2.4.4.2.34.2 Effect on Receipt

The receiver of this command learns the result of a previous ZDO Clear_All_Bindings_req.

2.4.4.3 Network Management Server Services

Table 2-100 lists the commands supported by Device Profile: Network Management Server Services. Each of these commands will be discussed in the following sections.

Table 2-100. Network Management Server Service Commands

Network Management Server Service Commands	Cluster ID	Server Processing
Mgmt_NWK_Disc_rsp	0x8030	Deprecated
Mgmt_Lqi_rsp	0x8031	M

Network Management Server Service Commands	**Cluster ID**	**Server Processing**
Mgmt_Rtg_rsp	0x8032	O
Mgmt_Bind_rsp	0x8033	O
Mgmt_Leave_rsp	0x8034	O
Mgmt_Direct_Join_rsp	0x8035	Deprecated
Mgmt_Permit_Joining_rsp	0x8036	M
Mgmt_Cache_rsp	0x8037	Deprecated
Mgmt_NWK_Update_notify	0x8038	O
Mgmt_NWK_Enhanced_Update_notify	0x8039	O
Mgmt_NWK_IEEE_Joining_List_rsp	0x803A	O
Mgmt_NWK_Unsolicited_Enhanced_Update_notify	0x803B	O
Mgmt_NWK_Beacon_Survey_rsp	0x803C	O

2.4.4.3.1 Mgmt_NWK_Disc_rsp – DEPRECATED COMMAND

2.4.4.3.2 Mgmt_Lqi_rsp

The Mgmt_Lqi_rsp command (ClusterID=0x8031) SHALL be formatted as illustrated in Figure 2-68.

Octets: 1	**1**	**1**	**1**	**Variable**
Status	NeighborTable Entries	Start Index	NeighborTable ListCount	NeighborTable List

Figure 2-68. Format of the Mgmt_Lqi_rsp Command Frame

Table 2-101 specifies the fields of the Mgmt_Lqi_rsp command frame.

Table 2-101. Fields of the Mgmt_Lqi_rsp Command Frame

Name	**Type**	**Valid Range**	**Description**
Status	Integer	NOT_SUPPORTED or any status code returned from the NLME-GET.confirm primitive.	The status of the Mgmt_Lqi_req command.

Name	Type	Valid Range	Description
NeighborTableEntries	Integer	0x00 – 0xff	Total number of Neighbor Table entries with unique addresses within the Remote Device.
StartIndex	Integer	0x00 – 0xff	Starting index within the Neighbor Table filtered on unique addresses to begin reporting for the NeighborTableList.
NeighborTableListCount	Integer	0x00 – 0x02	Number of Neighbor Table entries included within NeighborTableList.
NeighborTableList	List of Neighbor Descriptors	The list SHALL contain the number elements given by the NeighborTableListCount.	A list of descriptors, beginning with the StartIndex element and continuing for NeighborTableListCount, of the elements in the Remote Device's Neighbor Table including the device address and associated LQI (see Table 2-102 for details).

Table 2-102. NeighborTableList Record Format

Name	Size (Bits)	Valid Range	Description
Extended PAN Id	64	A 64-bit PAN identifier	The 64-bit extended PAN identifier of the neighboring device.
Extended address	64	An extended 64-bit, IEEE address	64-bit IEEE address that is unique to every device. If this value is unknown at the time of the request, this field SHALL be set to 0xffffffffffffffff.
Network address	16	Network address	The 16-bit network address of the neighboring device.
Device type	2	0x00 – 0x03	The type of the neighbor device: 0x00 = Zigbee coordinator 0x01 = Zigbee router 0x02 = Zigbee end device 0x03 = Unknown

Name	Size (Bits)	Valid Range	Description
RxOnWhenIdle	2	0x00 – 0x02	Indicates if neighbor's receiver is enabled during idle portions of the CAP: 0x00 = Receiver is off 0x01 = Receiver is on 0x02 = unknown
Affinity	3	0x00 – 0x03	The relationship between the neighbor and the current device: 0x00 = neighbor is the parent 0x01 = neighbor is a child 0x02 = neighbor is a sibling 0x03 = None of the above
Reserved	1		This reserved bit SHALL be set to 0.
Permit joining	2	0x00 - 0x02	An indication of whether the neighbor device is accepting join requests: 0x00 = neighbor is not accepting join requests 0x01 = neighbor is accepting join requests 0x02 = unknown
Reserved	6		Each of these reserved bits SHALL be set to 0.
Depth	8	0x00 – nwkcMaxDepth	The tree depth of the neighbor device. A value of 0x00 indicates that the device is the Zigbee coordinator for the network.
LQA	8	0x00 – 0xff	The estimated link quality for RF transmissions from this device. See section 3.6.3 for a discussion of how this is calculated.

2.4.4.3.2.1 When Generated

The Mgmt_Lqi_rsp is generated in response to an Mgmt_Lqi_req. If this management command is not supported, a status of NOT_SUPPORTED SHALL be returned and all parameter fields after the Status field SHALL be omitted. Otherwise, the Remote Device SHALL implement the following processing.

Upon receipt of and after support for the Mgmt_Lqi_req has been verified, the Remote Device SHALL perform an NLME-GET.request (for the *nwkNeighborTable* attribute) and process the resulting neighbor table (obtained via the NLME-GET.confirm primitive) to create the Mgmt_Lqi_rsp command. If *nwkNeighborTable* was successfully obtained but one or more of the fields required in the NeighborTableList record (see Table 2-102) are not supported (as they are optional), the Mgmt_Lqi_rsp SHALL return a status of NOT_SUPPORTED and all parameter fields after the Status field SHALL be omitted. Otherwise, the Mgmt_Lqi_rsp command SHALL contain the same status that was contained in the NLME-GET.confirm primitive and if this was not SUCCESS, all parameter fields after the status field SHALL be omitted.

The Relationship field in the nwkNeighborTable entry maps to the Affinity field in the Mgmt_Lqi_rsp but with the following special processing. Routers SHALL report back the Relationship status in the Affinity field as follows. If the Relationship enumeration is 0x00 to 0x02, then the Affinity field SHALL be the same value. If the Relationship enumeration indicates 0x03 or greater, then the Affinity field SHALL be set to 0x03, None of the Above.

From the *nwkNeighborTable* attribute, the neighbor table SHALL be accessed, starting with the index specified by StartIndex, and SHALL be moved to the NeighborTableList field of the Mgmt_Lqi_rsp command. The entries reported from the neighbor table SHALL be those, starting with StartIndex and including whole NeighborTableList records (see Table 2-102) until the limit on MSDU size, i.e., *aMaxMACFrameSize* (see [B1]), is reached. Within the Mgmt_Lqi_rsp command, the NeighborTableEntries field SHALL represent the total number of Neighbor Table entries in the Remote Device. The parameter NeighborTableListCount SHALL be the number of entries reported in the NeighborTableList field of the Mgmt_Lqi_rsp command.

The extended address, device type, RxOnWhenIdle, and permit joining fields have “unknown” values which SHALL be returned where the values are not available.

2.4.4.3.2.2 Effect on Receipt

The local device is notified of the results of its attempt to obtain the neighbor table.

2.4.4.3.3 Mgmt_Rtg_rsp

The Mgmt_Rtg_rsp command (ClusterID=0x8032) SHALL be formatted as illustrated in Figure 2-69.

Octets: 1	**1**	**1**	**1**	**Variable**
Status	RoutingTable Entries	Start Index	RoutingTable ListCount	RoutingTable List

Figure 2-69. Format of the Mgmt_Rtg_rsp Command Frame

Table 2-103 specifies the fields of the Mgmt_Rtg_rsp command frame.

Table 2-103. Fields of the Mgmt_Rtg_rsp Command Frame

Name	**Type**	**Valid Range**	**Description**
Status	Integer	NOT_SUPPORTED or any status code returned from the NLME-GET.confirm primitive.	The status of the Mgmt_Rtg_req command.
RoutingTableEntries	Integer	0x00– 0xff	Total number of Routing Table entries within the Remote Device.
StartIndex	Integer	0x00– 0xff	Starting index within the Routing Table to begin reporting for the RoutingTable-List.
RoutingTableListCount	Integer	0x00– 0xff	Number of Routing Table entries included within RoutingTableList.

Name	**Type**	**Valid Range**	**Description**
RoutingTableList	List of Routing Descriptors	The list SHALL contain the number elements given by the RoutingTableListCount	A list of descriptors, beginning with the StartIndex element and continuing for RoutingTableListCount, of the elements in the Remote Device's Routing Table (see Table 2-104 for details).

Table 2-104. RoutingTableList Record Format

Name	**Size (Bits)**	**Valid Range**	**Description**
Destination address	16	The 16-bit network address of this route.	Destination address.
Status	3	The status of the route.	0x0=ACTIVE. 0x1=DISCOVERY_UNDERWAY. 0x2=DISCOVERY_FAILED. 0x3=INACTIVE. 0x4-0x7=Reserved.
Memory Constrained	1		A flag indicating whether the device is a memory constrained concentrator.
Many-to-one	1		A flag indicating that the destination is a concentrator that issued a many-to-one request.
Route record required	1		A flag indicating that a route record command frame SHOULD be sent to the destination prior to the next data packet.
Reserved	2		
Next-hop address	16	The 16-bit network address of the next hop on the way to the destination.	Next-hop address.

2.4.4.3.3.1 When Generated

The Mgmt_Rtg_rsp is generated in response to an Mgmt_Rtg_req. If this management command is not supported, a status of NOT_SUPPORTED SHALL be returned and all parameter fields after the Status field SHALL be omitted. Otherwise, the Remote Device SHALL implement the following processing.

Upon receipt of and after support for the Mgmt_Rtg_req has been verified, the Remote Device SHALL perform an NLME-GET.request (for the *nwkRouteTable* attribute) and process the resulting NLME-GET.confirm (containing the *nwkRouteTable* attribute) to create the Mgmt_Rtg_rsp command. The Mgmt_Rtg_rsp command SHALL contain the same status that was contained in the NLME-GET.confirm primitive and if this was not SUCCESS, all parameter fields after the status field SHALL be omitted.

From the *nwkRouteTable* attribute, the routing table SHALL be accessed, starting with the index specified by StartIndex, and moved to the RoutingTableList field of the Mgmt_Rtg_rsp command. The entries reported from the routing table SHALL be those, starting with StartIndex and including whole RoutingTableList records (see Table 2-104) until MSDU size limit, that is, *aMaxMACFrameSize* (see [B1]), is reached. Within the Mgmt_Rtg_rsp command, the RoutingTableEntries field SHALL represent the total number of Routing Table entries in the Remote Device. The RoutingTableListCount field SHALL be the number of entries reported in the RoutingTableList field of the Mgmt_Rtg_req command.

2.4.4.3.3.2 Effect on Receipt

The local device is notified of the results of its attempt to obtain the routing table.

2.4.4.3.4 Mgmt_Bind_rsp

The Mgmt_Bind_rsp command (ClusterID=0x8033) SHALL be formatted as illustrated in Figure 2-70.

Octets: 1	**1**	**1**	**1**	**Variable**
Status	BindingTable Entries	Start Index	BindingTable ListCount	BindingTable List

Figure 2-70. Format of the Mgmt_Bind_rsp Command Frame

Table 2-105 specifies the fields of the Mgmt_Bind_rsp command frame.

Table 2-105. Fields of the Mgmt_Bind_rsp Command Frame

Name	**Type**	**Valid Range**	**Description**
Status	Integer	NOT_SUPPORTED or any status code returned from the APSME-GET.confirm primitive.	The status of the Mgmt_Bind_req command.
BindingTableEntries	Integer	0x00 – 0xff	Total number of Binding Table entries within the Remote Device.
StartIndex	Integer	0x00 – 0xff	Starting index within the Binding Table to begin reporting for the BindingTableList.
BindingTableListCount	Integer	0x00 – 0xff	Number of Binding Table entries included within BindingTableList.

Name	Type	Valid Range	Description
BindingTableList	List of Binding Descriptors	The list SHALL contain the number elements given by the BindingTableListCount.	A list of descriptors, beginning with the StartIndex element and continuing for BindingTableListCount, of the elements in the Remote Device's Binding Table (see Table 2-106 for details).

Table 2-106. BindingTableList Record Format

Name	Size (Bits)	Valid Range	Description
SrcAddr	64	A valid 64-bit IEEE address	The source IEEE address for the binding entry.
SrcEndpoint	8	0x01 – 0xfe	The source endpoint for the binding entry.
ClusterId	16	0x0000 – 0xffff	The identifier of the cluster on the source device that is bound to the destination device.
DstAddrMode	8	0x00 – 0xff	The addressing mode for the destination address. This field can take one of the non-reserved values from the following list: 0x00 = reserved 0x01 = 16-bit group address for DstAddr and DstEndpoint not present 0x02 = reserved 0x03 = 64-bit extended address for DstAddr and DstEndp present 0x04 – 0xff = reserved
DstAddr	16/64	As specified by the DstAddrMode field.	The destination address for the binding entry.
DstEndpoint	0/8	0x01 – 0xff	This field SHALL be present only if the DstAddrMode field has a value of 0x03 and, if present, SHALL be the destination endpoint for the binding entry.

2.4.4.3.4.1 When Generated

The Mgmt_Bind_rsp is generated in response to a Mgmt_Bind_req. If this management command is not supported, a status of NOT_SUPPORTED shall be returned and all parameter fields after the Status field shall be omitted. Otherwise, the Remote Device SHALL implement the following processing.

Upon receipt of and after support for the Mgmt_Bind_req has been verified, the Remote Device SHALL perform an APSME-GET.request (for the *apsBindingTable* attribute) and process the resulting APSME-GET.confirm (containing the *apsBindingTable* attribute) to create the Mgmt_Bind_rsp command. The Mgmt_Bind_rsp command SHALL contain the same status that was contained in the APSME-GET.confirm primitive and if this was not SUCCESS, all parameter fields after the status field SHALL be omitted. If the binding table is empty, the Mgmt_Bind_rsp SHALL return SUCCESS, set the fields BindingTable Entries = Start Index = BindingTable ListCount = 0x00 and not include the BindingTable List field.

From the *apsBindingTable* attribute, the binding table SHALL be accessed, starting with the index specified by StartIndex, and moved to the BindingTableList field of the Mgmt_Bind_rsp command. The entries reported from the binding table SHALL be those, starting with StartIndex and including whole BindingTableList records (see Table 2-106) until the MSDU size limit, that is,, *aMaxMACFrameSize* (see [B1]), is reached. Within the Mgmt_Bind_rsp command, the BindingTableEntries field SHALL represent the total number of Binding Table entries in the Remote Device. The BindingTableListCount field SHALL be the number of entries reported in the BindingTableList field of the Mgmt_Bind_req command.

2.4.4.3.4.2 Effect on Receipt

The local device is notified of the results of its attempt to obtain the binding table.

2.4.4.3.5 Mgmt_Leave_rsp

The Mgmt_Leave_rsp command (ClusterID=0x8034) SHALL be formatted as illustrated in Figure 2-71.

Octets: 1
Status

Figure 2-71. Format of the Mgmt_Leave_rsp Command Frame

Table 2-107 specifies the fields of the Mgmt_Leave_rsp command frame.

Table 2-107. Fields of the Mgmt_Leave_rsp Command Frame

Name	**Type**	**Valid Range**	**Description**
Status	Integer	NOT_SUPPORTED, NOT_AUTHORIZED or any status code returned from the NLME-LEAVE.confirm primitive.	The status of the Mgmt_Leave_req command.

2.4.4.3.5.1 When Generated

The Mgmt_Leave_rsp is generated in response to a Mgmt_Leave_req. Stacks certified prior to Revision 21 MAY or MAY NOT support this command. If this management command is not supported, a status of NOT_SUPPORTED SHALL be returned. All stacks certified to Revision 21 and later SHALL support this command.

2.4.4.3.5.2 Effect on Receipt

Upon receipt of the Mgmt_leave_rsp the device MAY parse the Status field to determine whether or not the remote device accepted the leave request.

2.4.4.3.6 Mgmt_Direct_Join_rsp – DEPRECATED

2.4.4.3.7 Mgmt_Permit_Joining_rsp

The Mgmt_Permit_Joining_rsp command (ClusterID=0x8036) SHALL be formatted as illustrated in Figure 2-72.

Octets: 1
Status

Figure 2-72. Format of the Mgmt_Permit_Joining_rsp Command Frame

Table 2-108 specifies the fields of the Mgmt_Permit_Joining_rsp command frame.

Table 2-108. Fields of the Mgmt_Permit_Joining_rsp Command Frame

Name	**Type**	**Valid Range**	**Description**
Status	Integer	SUCCESS, INV_REQUESTTYPE, NOT_AUTHOR-IZED, or any status code returned from the NLME-PERMIT-JOINING.confirm primitive.	The status of the Mgmt_Permit_Joining_rsp command.

2.4.4.3.7.1 When Generated

The Mgmt_Permit_Joining_rsp is generated in response to a unicast Mgmt_Permit_Joining_req. In the description which follows, note that no response SHALL be sent if the Mgmt_Permit_Joining_req was received as a broadcast to all routers. If this management command is not permitted by the requesting device, a status of INV_REQUESTTYPE SHALL be returned. Upon receipt and after support for Mgmt_Permit_Joining_req has been verified, the Remote Device SHALL execute the NLME-PERMIT-JOINING.request. The Mgmt_Permit-Joining_rsp SHALL contain the same status that was contained in the NLME-PERMIT-JOINING.confirm primitive.

2.4.4.3.7.2 Effect on Receipt

The status of the Mgmt_Permit_Joining_req command is notified to the requestor.

2.4.4.3.8 Mgmt_Cache_rsp – DEPRECATED

2.4.4.3.9 Mgmt_NWK_Update_notify

The Mgmt_NWK_Update_notify command (ClusterID=0x8038) SHALL be formatted as illustrated in Figure 2-73.

Octets: 1	**4**	**2**	**2**	**1**	**Variable**
Status	Scanned Channels	TotalTransmis-sions	Transmission-Failures	ScannedChan-nelsListCount	EnergyValues

Figure 2-73. Format of the Mgmt_NWK_Update_notify Command Frame

Table 2-109 specifies the fields of the Mgmt_NWK_Update_notify command frame.

Table 2-109. Fields of the Mgmt_NWK_Update_notify Command Frame

Name	Type	Valid Range	Description
Status	Integer	SUCCESS, INV_REQUESTTYPE, NOT_SUPPORTED, or any status values returned from the MLME-SCAN.confirm primitive	The status of the Mgmt_NWK_Update_notify command.
ScannedChannels	Bitmap	0x00000000 – 0xffffffff.	The five most significant bits (b27,..., b31) represent the binary encoded Channel Page. The 27 least significant bits (b0, b1,... b26) indicate which channels were scanned (1 = scan, 0 = do not scan) for each of the 27 valid channels.
TotalTransmissions	Integer	0x0000 –0xffff	Count of the total transmissions reported by the device.
TransmissionFailures	Integer	x0000 –0xffff	Sum of the total transmission failures reported by the device.
ScannedChannelsList-Count	Integer	0x00 – 0xff	The list SHALL contain the number of records contained in the EnergyValues parameter.
EnergyValues	Integer	List of ED values each of which can be in the range of 0x00 – 0xff.	The result of an energy measurement made on this channel in accordance with [B1].

2.4.4.3.9.1 When Generated

The Mgmt_NWK_Update_notify is provided to enable Zigbee devices to report the condition on local channels to a network manager. The scanned channels list is the report of channels scanned and contains a count followed by a list of records, one for each channel scanned, each record including one byte of the energy level measured during the scan, or 0xff if there is too much interference on this channel.

When sent in response to a Mgmt_NWK_Update_req command the status field SHALL represent the status of the request. This message SHALL NOT be sent unsolicited – use Mgmt_NWK_Unsolicited_Enhanced_Update_notify instead.

2.4.4.3.9.2 Effect on Receipt

The local device is notified of the local channel conditions at the transmitting device, or of its attempt to update network configuration parameters.

2.4.4.3.10 Mgmt_NWK_Enhanced_Update_notify

The Mgmt_NWK_Enhanced_Update_notify command (ClusterID=0x8039) SHALL be formatted as illustrated in Figure 2-74.

Octets: 1	4	2	2	1	Variable
Status	Scanned Channels	TotalTransmis-sions	Transmission-Failures	ScannedChan-nelsListCount	EnergyValues

Figure 2-74. Format of the Mgmt_NWK_Enhanced_Update_notify Command Frame

Table 2-110 specifies the fields of the Mgmt_NWK_Enhanced_Update_notify command frame.

Table 2-110. Fields of the Mgmt_NWK_Enhanced_Update_notify Command Frame

Name	Type	Valid Range	Description
Status	Integer	SUCCESS, INV_REQUESTTYPE, NOT_SUPPORTED, or any status values returned from the MLME-SCAN.confirm primitive.	The status of the Mgmt_NWK_Enhanced_Update_notify command.
ScannedChannels	Bitmap	0x00000000 – 0xffffffff.	The five most significant bits (b27,..., b31) represent the binary encoded Channel Page. The 27 least significant bits (b0, b1,... b26) indicate which channels were scanned (1 = scan, 0 = do not scan) for each of the 27 valid channels.
TotalTransmissions	Integer	0x0000 – 0xffff	Count of the total transmissions reported by the device.
TransmissionFailures	Integer	x0000 – 0xffff	Sum of the total transmission failures reported by the device.
ScannedChannelsList-Count	Integer	0x00 – 0xff	The list SHALL contain the number of records contained in the EnergyValues parameter.
EnergyValues	Integer	List of ED values each of which can be in the range of 0x00 – 0xff.	The result of an energy measurement made on this channel in accordance with [B1].

2.4.4.3.10.1 When Generated

The Mgmt_NWK_Enhanced_Update_notify is provided to enable Zigbee devices to report the condition on local channels to a network manager. The scanned channels list is the report of channels scanned and contains a count followed by a list of records, one for each channel scanned, each record including one byte of the energy level measured during the scan, or 0xff if there is too much interference on this channel.

When sent in response to a Mgmt_NWK_Enhanced_Update_req command the status field SHALL represent the status of the request. This message SHALL NOT be sent unsolicited – use Mgmt_NWK_Unsolicited_Enhanced_Update_notify instead.

4736 ##### 2.4.4.3.10.2 Effect on Receipt

4737 The local device is notified of the local channel conditions at the transmitting device, or of its attempt to update
4738 network configuration parameters.

4739 #### 2.4.4.3.11 Mgmt_NWK_IEEE_Joining_List_rsp

4740 The Mgmt_NWK_IEEE_Joining_list_rsp command (Cluster ID=0x803A) SHALL be formatted as illustrated in Fig-
4741 ure 2-75.

Octets: 1	0/1	0/1	0/1	0/1	0/1	0/Variable
Status	IeeeJoin-ingListUpdateID	JoiningPol-icy	IeeeJoiningListTotal	StartIn-dex	IeeeJoiningCount	IeeeJoin-ingList

4742 **Figure 2-75. Format of the Mgmt_NWK_IEEE_Joining_List_rsp Command Frame**

4743 Table 2-111 specifies the fields of the Mgmt_NWK_IEEE_Joining_List_rsp command frame.

4744 **Table 2-111. Field Descriptions of the Mgmt_NWK_IEEE_Joining_List_rsp Command Frame**

Name	Type	Valid Range	Description
Status	Integer	SUCCESS, INV_REQUESTTYPE, or NOT_SUPPORTED	The status of the Mgmt_NWK_IEEE_Joining_List_req command. If Status is not SUCCESS, no other fields are included.
IeeeJoiningListUpdateID	Integer	0x00 – 0xFF	The issue ID of the IeeeJoiningList. This field SHALL start at 0 and increment for each change to the IeeeJoiningList, or each change to the Joining Policywrapping to 0 after 0xFF.
JoiningPolicy	Enumeration	See Table 2-112.	This is an enumeration indicating one of the JoiningPolicy values allowed in Table 2-112.
IeeeJoiningListTotal	Integer	0x00 – 0xFF	The total number of IEEE Joining Addresses contained in the Mgmt_NWK_IEEE_Joining_List_rsp.
StartIndex	Integer	0x00 – 0xFF	The starting index in the mibIeeeJoiningList. This field SHALL be omitted if the IeeeJoiningListTotal is 0.
IeeeJoiningCount	Integer	x00 – 0xFF	The number of IEE joining messages contained in the ZDO message
IeeeJoiningList	List of IEEE values		A list of IEEE addresses from the mibIeeeJoiningList. This field SHALL be omitted if the IeeeJoiningListTotal is 0.

Table 2-112. ZDO JoiningPolicy Enumeration Values

Enumeration	Value	Description
ALL_JOIN	0x00	Any device is allowed to join.
IEEELIST_JOIN	0x01	Only devices on the mibJoiningIeeeList are allowed to join.
NO_JOIN	0x02	No device is allowed to join.

2.4.4.3.11.1 When Generated

The Mgmt_NWK_IEEE_Joining_List_rsp MAY either be generated in response to a Mgmt_NWK_IEEE_Joining_List_req or it MAY be sent as an unsolicited broadcast to inform the entire network of a change. For the details of when it is generated in response to a Mgmt_NWK_IEEE_Joining_List_req, see section 2.4.3.3.11.2.

2.4.4.3.11.2 Effect on Receipt

The device SHALL process the message as follows:

1) If the Status is not SUCCESS, the message SHALL be discarded and no further processing SHALL take place.
2) For each entry in the nwkMacInterfaceTable it SHALL do the following.
   a) Execute an MLME-SET.request of the *mibJoiningPolicy* to the value of the JoiningPolicy from the ZDO message.
   b) If the IeeeJoiningListTotal is 0 it SHALL do the following:
      i) The ZDO SHALL clear all entries from the *mibJoiningIeeeList*.
      ii) Go to step 2 and process the next entry in the nwkMacInterfaceTable.
   c) Execute an MLME-SET.request and set the values of the *mibJoiningIeeeList* at the index of StartIndex to the values of IeeeJoiningList from the ZDO message.

2.4.4.3.12 Mgmt_NWK_Unsolicited_Enhanced_Update_notify

The Mgmt_NWK_Unsolicited_Enhanced_Update_notify command (ClusterID=0x003b) SHALL be formatted as illustrated in Figure 2-76.

Octets: 1	4	2	2	2	1
Status	Channel in use	MACTxUcast Total	MACTxUcast Failures	MACTxUcast Retries	PeriodOfTimeForResults

Figure 2-76. Format of the Mgmt_NWK_Unsolicited_Update_notify Command Frame

Table 2-113 specifies the fields of the Mgmt_NWK_Unsolicited_Enhanced_Update_notify command frame.

Table 2-113 Fields of the Mgmt_NWK_Unsolicited_Enhanced_Update_notify Command

Name	Type	Valid Range	Description
Channel in use	Bitmap	0x00000000 – 0xffffffff	The five most significant bits (b27,..., b31) represent the binary encoded Channel Page. The 27 least significant bits (b0, b1,... b26) indicate which channels

Name	Type	Valid Range	Description
			is in use (1 = in use, 0 = not in use) for each of the 27 valid channels.
MACTxUcast Total	Integer	0x0000 –0xffff	Total number of Mac Tx Transactions to attempt to send a message (but not counting retries)
MACTxUcast Failures	Integer	x0000 – 0xffff	Total number of failed Tx Transactions. So if the Mac sent a single packet, it will be retried 4 times without ACK, that counts as 1 failure.
MACTxUcast Retries	Integer	x0000 – 0xffff	Total number of Mac Retries regardless of whether the transaction resulted in success or failure.
PeriodOfTimeForResults	Integer	0x00 – 0xff	Time period over which MACTxyyy results are measured (in minutes)

2.4.4.3.12.1 When Generated

The Mgmt_NWK_Unsolicited_Enhanced_Update_notify is provided to enable Zigbee devices to report the condition on local channels to a network manager. The scanned channel list is the report of channels scanned and it is followed by a list of records, one for each channel scanned, each record including one byte of the energy level measured during the scan, or 0xff if there is too much interference on this channel.

2.4.4.3.12.2 Effect on Receipt

The local device is notified of the local channel conditions at the transmitting device.

2.4.4.3.13 Mgmt_NWK_Beacon_Survey_rsp

The Mgmt_NWK_Beacon_Survey_rsp (ClusterID=0x803c) SHALL be formatted as illustrated in Figure 2-77.

Octets: 1	Varies
Status	TLVs

Figure 2-77. Format of the Mgmt_NWK_Beacon_Survey_rsp Command Frame

Table 2-114 specifies the fields of the Mgmt_NWK_Beacon_Survey_rsp command frame.

Table 2-114. Fields of the Mgmt_NWK_Beacon_Survey_rsp Command Frame

Name	Type	Valid Range	Description
Status	Integer	SUCCESS, INV_REQUESTTYPE, or NOT_SUPPORTED	The status of the Mgmt_NWK_Beacon_Survey_req command. If the status is not SUCCESS, then the other fields are not included.
TLVs	TLV	Varies	The Mgmt_NWK_Beacon_Survey_rsp SHALL include the following TLVs:

			• Beacon Survey Results TLV • Potential Parents TLV

2.4.4.3.13.1 Local TLVs

2.4.4.3.13.1.1 Beacon Survey Results TLV (ID 0x01)

The Beacon Survey Results TLV (ID 0x01) is 4 bytes in length and contains information about the channels, scan configuration and counted beacons as illustrated in Figure 2-78.

Octets: 1	1	1	1
Total Beacons Received	On Network Beacon	Potential Parent Beacon	Other Network Beacons

Figure 2-78. Format of the Beacon Survey Results TLV

Table 2-115 specifies the fields of the Beacon Survey Results TLV.

Table 2-115. Fields of the Beacon Survey Results TLV

Name	Type	Valid Range	Description
Total Beacons Received	Integer	0 – 255	The total number of IEEE Std 802.15.4 beacons received during the scan.
On-network Beacons	Integer	0 – 255	The total number of Zigbee Network beacons where the Extended PAN ID matches the local device's nwkExtendedPanId.
Potential Parent Beacons	Integer	0 – 255	The total number of Zigbee Network beacons where the Extended PAN ID matches and the Zigbee Beacon payload indicates End Device Capacity = TRUE.
Other Network Beacons	Integer	0 – 255	The total number of IEEE Std 802.15.4 beacons from other Zigbee networks or other IEEE Std 802.15.4 networks. Other Zigbee network beacons are defined as when the Extended PAN ID does not match the local Extended PAN ID.

2.4.4.3.13.1.2 Potential Parents TLV(ID 0x02)

The Potential Parents TLV(ID 0x02) is 4 to 19 bytes in length and indicates the number of available parents in radio range as illustrated in Figure 2-79. A maximum of 5 parents is supported for this TLV. The list of potential parents SHALL be ordered as described in section 3.6.1.5.2.

Octets: 2	1	1	0 / 2	0 / 1	Variable
Current Parent Short Address	LQA	Count	Potential Parent Short Address	LQA	Additional Potential Parent Short Address and LQA fields

Figure 2-79. Format of the Potential Parents TLV

Table 2-116 specifies the fields of the Potential Parents TLV.

Table 2-116. Fields of the Potential Parents TLV

Name	Type	Valid Range	Description
Current Parent Short Address	Short Address	0x0000 – 0xFFFF	The short address that is the current parent for the device. For a router or coordinator this value SHALL be set to 0xFFFF.
LQA	Integer	0x00 – 0xFF	The value of the LQA of the current parent.
Count	Integer	0x00 – 0x05	This is the count of additional potential parent short addresses and their associated LQA. If there are no other potential parents this SHALL indicate 0. This value SHALL not be greater than 5.
Potential Parent Short Address	Short Address	0x0000 – 0xFFFF	The short address for a potential parent that the device can hear a beacon for.
LQA	Integer	0x00 – 0xFF	The LQA value of the associated potential parent.

2.4.4.3.13.2 When Generated

This is generated in response to the Mgmt_NWK_Beacon_Survey_req command.

2.4.4.3.13.3 Effect on Receipt

The application MAY use this to help manage the network.

2.4.4.4 Security Server Services

Table 2-117 lists the commands supported by the Device Profile related to Security Client services.

Table 2-117. Security Server Services

Security Client Service	Cluster ID	Client Transmission	Server Processing
Security_Start_Key_Negotiation_rsp	0x8040	O	O
Security_Retrieve_Authentication_Token_rsp	0x8041	O	O
Security_Get_Authentication_Level_rsp	0x8042	M	M
Security_Set_Configuration_rsp	0x8043	M	M
Security_Get_Configuration_rsp	0x8044	M	M
Security_Start_Key_Update_rsp	0x8045	M	M
Security_Decommisioning_rsp	0x8046	M	M
Security_Challenge_rsp	0x8047	M	M

2.4.4.4.1 Security_Start_Key_Negotiation_rsp

The Security_Start_Key_Negotiation_rsp command (0x8040) shall be formatted as illustrated in Figure 2-80. This command SHALL NOT be APS encrypted.

When performing Key Negotiation with an unauthenticated neighbor that is not yet on the network, network layer encryption SHALL NOT be used on the message. If the message is being sent to unauthenticated device that is not on the network and is not a neighbor, it SHALL be relayed as described in section 4.6.3.7.7. Otherwise the message SHALL have network layer encryption.

Octets: 1	Variable
Status	TLVs

Figure 2-80. Format of the Security_Start_Key_Negotiation_rsp Command Frame

Table 2-118 specifies the fields of the Security_Start_Key_Negotation_rsp command frame.

Table 2-118. Fields of the Security_Start_Key_Negotation_rsp Command Frame

Name	Type	Valid Range	Description
Status	Integer	SUCCESS, INVALID_TLV, MISSING_TLV, TEMPORARY_FAILURE, NOT_AUTHORIZED	The result of the Security_Start_Key_Negotiation_req.
TLVs	TLV	Varies	The set of TLVs sent by the receiver of the Security_Start_Key_Negotiation_req.

2.4.4.4.1.1 Local TLVs

2.4.4.4.1.2 Curve25519 Public Point TLV (ID=0)

Figure 2-81 indicates the format of the Local TLV for Curve25519 Public Point TLV.

Octets: 8	32
Device EUI64	Public Point

Figure 2-81. Format of the Curve25519 Public Point TLV

Table 2-119 specifies the fields of the Curve25519 Public Point TLV

Table 2-119. Fields of the Curve25519 Public Point TLV

Field	Description
Device EUI64	This indicates the EUI64 of the device that generated the public point.
Public Point	The 32-byte Curve public point.

2.4.4.4.1.3 When Generated

The Security_Start_Key_Negotiation_rsp is generated after a device processes the Security_Start_Key_Negotiation_req and decides to reject the request, or after it has accepted the request and executed the corresponding cryptographic primitives. Typically, this is used to negotiate a Trust Center Link Key prior to becoming fully joined and authorized on a network, but it can be used after joining a network as well.

The security primitives for key negotiation are the APSME-KEY-NEGOTIATION primitives and are used by the stack to manage the process. See section 4.4.9 for more details. Their interaction with the over-the-air messages can be found in Figure 4-6.

When negotiating a Trust Center Link Key the device SHALL send at least the following TLV:

- Curve25519 Public Point TLV

2.4.4.4.1.4 Effect on Receipt

On receipt, the device SHALL do as follows:

1. If the Status is TEMPORARY_FAILURE, indicating that the current APSME-KEY-NEGOTIATION.request cannot be processed at the present time, the Stack SHOULD retry the operation by generating a new APSME-KEY-NEGOTIATION.request. The delay before initiating the retry SHALL be 5 seconds or greater.
2. If the Status is any other non-zero value then no further processing SHALL be done.
3. If more than one public point TLV is present then the message SHALL be dropped and no further processing SHALL be done.
4. If the Curve25519 Public Point TLV is not present, then the message SHALL be dropped and no more processing SHALL be done.
5. Generate an APSME-KEY-NEGOTIATE.confirm with the following parameters
   a. The PartnerLongAddress SHALL be set to the Device EUI64 within the Curve25519 Public Point TLV.
   b. The PublicPointData SHALL be set to the public point from the Curve25519 Public Point TLV.
   c. If the ZDO frame was contained within an APS Command Relay Message Upstream then it SHALL do the following:
      i. Set RelayCommand to TRUE.
      ii. Set RelayLongAddress to the address of the Device that sent the Network Data frame.

2.4.4.4.2 Security_Retrieve_Authentication_Token_rsp

The Security_Retrieve_Authentication_Token_rsp command SHALL be as illustrated in Figure 2-82.

Octets: 1	Variable
Status	TLVs

Figure 2-82. Format of the Security_Retrieve_Authentication_Token_rsp Command Frame

2.4.4.4.2.1 When Generated

This message is generated by the Trust Center as described in section 2.4.3.4.2.

2.4.4.4.2.2 Effect on Receipt

Upon receipt, the device SHALL do the following:

1. If the message was not APS encrypted by the Trust Center it SHALL be dropped and no further processing SHALL be done.
2. If the message was not sent by the Trust Center it SHALL be dropped and no further processing SHALL be done.
3. The device SHALL find the *apsDeviceKeyPairSet* entry associated with the Trust Center.
   a. If none is found, then the message SHALL be discarded and no further processing SHALL be done.
4. The device SHALL examine the PassphraseUpdateAllowed of the entry.
   a. If set to FALSE then the message SHALL be discarded and no further processing SHALL be done.
5. The device SHALL examine the TLVs and determine if there is a 128-bit Symmetric Passphrase Global TLV in the set.
   a. If none is present, then the message SHALL be discarded and no further processing SHALL be done.
6. The device SHALL copy the data of the 128-bit Symmetric Passphrase Global TLV to the value of the Passphrase value for the entry of the *apsDeviceKeyPairSet* AIB value.
7. The device SHALL set the PassphraseUpdateAllowed value of the entry to FALSE.

2.4.4.4.3 Security_Get_Authentication_Level_rsp

The Security_Get_Authentication_Level_rsp command (ClusterID= 0x8042) SHALL be formatted as illustrated in Figure 2-83.

Octets: 1	Variable
Status	TLVs

Figure 2-83. Format of the Security_Get_Authentication_Level_rsp Command Frame

Table 2-120 specifies the fields of the Security_Start_Key_Negotation_rsp command frame.

Table 2-120. Fields of the Security_Get_Authentication_Level_rsp Command Frame

Name	Type	Valid Range	Description
Status	Integer	SUCCESS, NOT_SUPPORTED, INV_REQUESTTYPE, MISSING_TLV, and NOT_AUTHORIZED	The status of the request to get the authentication level.
TLVs	TLVs	Varies	A list of one or more TLVs. The following TLVs have specified behavior in this Revision of the specification: • Device Authentication Level TLV Other TLVs may be included.

2.4.4.4.3.1 Local TLVs

2.4.4.4.3.2 Device Authentication Level TLV (ID=0)

The Device Authentication Level TLV is formatted as illustrated in Figure 2-84.

Octets: 8	1	1
IEEEAddrRemoteNode	InitialJoinMethod	AcitveLinkKeyType

Figure 2-84. Format of the Device Authentication TLV

Table 2-121 specifies the fields of the Device Authentication TLV.

Table 2-121. Fields of the Device Authentication TLV

Name	Type	Valid Range	Description
IEEEAddrRemoteNode	Device Address	An extended 64-bit, IEEE address	64-bit address for the node that is be-ing inquired about.
InitialJoinMethod	Enumeration	0x00 – 0x03	This indicates the joining method that was used when the device joined the network. 0x00 = Anonymous

Name	Type	Valid Range	Description
			0x01 = Install Code Key 0x02 = Well-known Passphrase 0x03 = Install Code Passphrase
ActiveLinkKeyType	Enumeration	0x00 – 0x04	This indicates what Link Key update method was used to create the current active Link Key. 0x00 = Not Updated 0x01 = Key Request Method 0x02 = Unauthenticated Key Negotiation 0x03 = Authenticated Key Negotiation 0x04 = Application Defined Certificate Based Mutual Authentication

2.4.4.4.3.3 When Generated

The Security_Get_Authentication_Level_rsp is generated by a Remote Device in response to a Security_Get_Authentication_Level_req command inquiring as to the authentication level of the IEEEAddrOfInterest of an address held in the Key Pair Descriptor table. The destination addressing on this command SHALL be unicast. The command SHALL be APS encrypted.

2.4.4.4.3.4 Effect on Receipt

On receipt of the Security_Get_Authentication_Level_rsp command, the recipient is either notified of the status of its attempt to discover the current authentication level of an IEEE address or notified of an error. If the Security_Get_Authentication_Level_rsp command is received with a Status of SUCCESS, the remaining fields of the command contain the appropriate discovery information.

2.4.4.4.4 Security_Set_Configuration_rsp

The Security_Set_Configuration_rsp command (ClusterID=0x8043) SHALL be formatted as illustrated in Figure 2-85. The command contains a set of TLV Tag ID and TLV Processing Status pairs as defined by the TLV Status Count in Figure 2-86.

Octets: 1	Variable
Overall Status	TLVs

Figure 2-85. Security_Set_Configuration_rsp Command Frame

Table 2-122 specifies the fields of the Security_Set_Configuration_rsp command frame.

Table 2-122. Fields of the Security_Set_Configuration_rsp Command Frame

Name	Type	Valid Range	Description
Overall Status	Integer	SUCCESS, INV_REQUESTTYPE, or NOT_SUPPORTED	The overall status of a Security_Set_Configuration_req command.
TLVs	Variable	Varies	A set of one or more TLVs.

2.4.4.4.4.1 Local TLVs

2.4.4.4.4.1.1 Processing Status TLV (ID = 0)

The Processing Status TLV indicates the result of processing configuration changes from a set of TLVs sent in a previous message. The Processing Status TLV illustrated in Figure 2-86 will be 1 or more bytes in length and contain pairs of tag ID and processing status results, meaning it will always be an odd number in total length.

Octets: 1	0 or 1	0 or 1	0 or 1	0 or 1	…
TLV Status Count	Tag ID	Processing Status	Tag ID	Processing Status	…

Figure 2-86. Format of the Processing Status TLV

The Processing Status TLV contains a set of Tag ID and Processing Status results from a previous set of TLVs sent to the device to change its configuration. The TLV Status count will indicate the number of Tag ID and Processing Status pairs are present in the full TLV. The count may be zero, indicating that there were no known TLVs in the previous message that could be processed. When the TLV Status count is greater than 1, there SHALL be pairs of Tag ID and Processing Status values. For each pair, the tag ID will indicate a previously received TLV tag ID and the associated status of whether it is processed. The Processing Status value SHALL be one of the ZDP Enumerated Status values: SUCCESS, INV_REQUESTTYPE, or NOT_SUPPORTED.

2.4.4.4.4.2 When Generated

The Security_Set_Configuration_rsp is generated by a device in response to a Security_Set_Configuration_req. For each received TLV Tag ID in the Security_Set_Configuration_req there SHALL exist a TLV Tag ID and the corresponding TLV Processing Status of that TLV. If at least one TLV was successfully processed the Overall Status SHALL be SUCCESS.

2.4.4.4.4.3 Effect on Receipt

The device receiving this message can determine the results of a previous Security_Set_Configuration_req.

2.4.4.4.5 Security_Get_Configuration_rsp

The Security_Get_Configuration_rsp command (ClusterID = 0x08044) is generated by a device in response to a Security_Get_Configuration_req. For received Global TLV IDs in the prior request the device responds with its current state information as a list of TLVs contained in the Security_Get_Configuration_rsp. This command SHALL be APS encrypted. The format of the message is in Figure 2-87.

Octets: 1	Variable
Overall Status	TLVs

Figure 2-87. Security_Get_Configuration_rsp Command Frame

Table 2-123 specifies the fields of the Security_Get_Configuration_rsp command frame.

Table 2-123. Fields of the Security_Get_Configuration_rsp Command Frame

Name	Type	Valid Range	Description
Overall Status	Integer	SUCCESS, INV_REQUESTTYPE, or NOT_SUPPORTED	The overall status of a Security_Get_Configuration_req command.
TLVs	TLV	Variable	The value of the requested global TLV values.

2.4.4.4.5.1 Local TLVs

There are no Local TLVs defined for this message.

2.4.4.4.5.2 When Generated

This message is generated in response to the ZDO Security_Get_Configuration_req.

2.4.4.4.5.3 Effect on Receipt

The device can examine each received global TLV to learn the state of that TLV for the device sending the Security_Get_Configuration_rsp.

2.4.4.4.6 Security_Start_Key_Update_rsp

The Security_Start_Key_Update_rsp command (cluster ID = 0x8045) is formatted as illustrated in Figure 2-88. This command SHALL be APS encrypted.

Octets: 1
Status

Figure 2-88. Security_Start_Key_Update_rsp Command Frame

Table 2-124 specifies the fields of the Security_Start_Key_Update_rsp command frame.

Table 2-124. Fields of the Security_Start_Key_Update_rsp Command Frame

Name	Type	Valid Range	Description
Status	Integer	SUCCESS, INV_REQUESTTYPE, NOT_AUTHORIZED or NOT_SUPPORTED	The status of the request to Start the key update process.

2.4.4.4.6.1 When Generated

This is generated in response to a Security_Start_Key_Update_req.

2.4.4.4.6.2 Effect on Receipt

The Trust Center will learn the result of whether it's request ZDO Security_Start_Key_Update_req was successful.

2.4.4.4.7 Security_Decommission_rsp

The Security_Decomission_rsp is sent in response to a Security_Decomission_req to report the result of an attempt to decomission all data associated with a target EUI64. The command (cluster ID = 0x8046) is formatted as illustrated in Figure 2-89. This command SHALL be APS encrypted.

Octets: 1
Status

Figure 2-89. Security_Decommission_rsp Command Frame

Table 2-125 specifies the fields of the Security_Decommission_rsp command frame.

Table 2-125. Fields of the Security_Decommission_rsp Command Frame

Name	Type	Valid Range	Description
Status	Integer	SUCCESS, INV_REQUESTTYPE, NOT_AUTHORIZED or NOT_SUPPORTED	The status of the request to Start the key update process.

2.4.4.4.7.1 When Generated

This is generated in response to a Security_Decommission_req.

2.4.4.4.7.2 Effect on Receipt

The Trust Center will learn of the result of the decommissioning request of a third-party device by the sender of the Security_Decommission_rsp.

2.4.4.4.8 Security_Challenge_rsp

This command is used by a device to respond to a challenge and send its latest frame counter value to another device. The Security_Challenge_rsp (Cluster ID = 0x8047) is formatted as illustrated in Figure 2-90.

Octets: Varies
TLVs

Figure 2-90. Security_Challenge_rsp Command Frame

2.4.4.4.8.1 Locally Scoped TLVs

Table 2-126 defines the locally scoped TLVs for this message.

Table 2-126. Locally Scoped TLVs for Security_Challenge_rsp

Tag ID	Name
0x00	APS Frame Counter Response

2.4.4.4.8.2 APS Frame Counter Response TLV

Table 2-127 describes the format of the APS Frame Counter Response TLV.

Table 2-127. Format of the APS Frame Counter Response TLV

Octets: 8	8	4	4	8
Sender EUI64	Received Challenge Value	APS Frame Counter	Challenge Security Frame Counter	MIC

Table 2-128 describes the fields of the APS Frame Counter Response TLV.

Table 2-128 Fields of the APS Frame Counter Response TLV

Field	Description
Responder EUI64	The EUI64 of the device that is responding to the Security_Challenge_req with its own challenge.
Received Challenge Value	A randomly generated 64-bit value previously received in the APS Frame Counter Challenge TLV.
APS Frame Counter	The current outgoing APS security frame counter held by the Responder EUI64 device.
Challenge Security Frame Counter	The AES-CCM-128 outgoing frame counter used to generate the MIC over the octet sequence { tag || length || responder EUI-64 || received challenge value || APS frame counter } using the special nonce and AES-128 key for frame counter synchronization.
MIC	The AES-128-CCM 64-bit MIC (security level 2) on all previous fields of this TLV, excluding the challenge security frame counter, including Tag ID and length fields.

2.4.4.4.8.3 When Generated

This command is generated by a device responding to a ZDO Security_Challenge_req to inform the requester of the local device's current APS Frame counter.

This command SHALL NOT be APS encrypted.

2.4.4.4.8.4 Effect on Receipt

1. If the message was broadcast it SHALL be dropped and no further processing SHALL be done.
2. If the message did not include the APS Frame Counter Response TLV do the following.
   a. The message is dropped and no further processing SHALL be done.
3. If the Sender EUI64 does not match the *apsChallengeTargetEui64* then the message SHALL be dropped and no further processing SHALL be done.
4. If the apsChallengeValue of the AIB does not match the Challenge Value in the TLV, the message SHALL be dropped and no further processing SHALL be done.
5. Otherwise, follow the procedure in section .

2.4.5 ZDP Enumeration Description

This section explains the meaning of the enumerations used in the ZDP. Table 2-129 shows a description of the ZDP enumeration values.

Table 2-129. ZDP Enumerations Description

Enumeration	Value	Description
SUCCESS	0x00	The requested operation or transmission was completed successfully.
-	0x01 – 0x7f	Reserved.
INV_REQUESTTYPE	0x80	The supplied request type was invalid.
DEVICE_NOT_FOUND	0x81	The requested device did not exist on a device following a child descriptor request to a parent.
INVALID_EP	0x82	The supplied endpoint was equal to 0x00 or 0xff.
NOT_ACTIVE	0x83	The requested endpoint is not described by a simple descriptor.
NOT_SUPPORTED	0x84	The requested optional feature is not supported on the target device.
TIMEOUT	0x85	A timeout has occurred with the requested operation.
NO_MATCH	0x86	failure to match any suitable clusters.
-	0x87	Reserved.
NO_ENTRY	0x88	The unbind request was unsuccessful due to the coordinator or source device not having an entry in its binding table to unbind.
NO_DESCRIPTOR	0x89	A child descriptor was not available following a discovery request to a parent.
INSUFFICIENT_SPACE	0x8a	The device does not have storage space to support the requested operation.
NOT_PERMITTED	0x8b	The device is not in the proper state to support the requested operation.
TABLE_FULL	0x8c	The device does not have table space to support the operation.
NOT_AUTHORIZED	0x8d	The device has rejected the command due to security restrictions.

Enumeration	Value	Description
DEVICE_BINDING_TABLE_FULL	0x8e	The device does not have binding table space to support the operation.
INVALID_INDEX	0x8f	The index in the received command is out of bounds.
FRAME_TOO_LARGE	0x90	The response was too large to fit in a single unfragmented message.
BAD_KEY_NEGOTIATION_METHOD	0x91	The requested Key Negotiation Method was not accepted.
TEMPORARY_FAILURE	0x92	The request encountered a temporary failure but a retry at a later time should be attempted and may succeed.
-	0x92 – 0xff	Reserved.

2.4.6 ZDP Enumeration Status Values from the Network Layer

The ZDP may reuse status values from the network layer according to the processing rules of the ZDO commands. One of the main uses of this will be to indicate INVALID_TLV or MISSING_TLV. This will avoid defining those status values at multiple layers. See Table 3-80 for the definition of those values.

2.4.7 Conformance

When conformance to this Profile is claimed, all capabilities indicated mandatory for this Profile SHALL be supported in the specified manner (process mandatory). This also applies to optional and conditional capabilities, for which support is indicated, and is subject to verification as part of the Zigbee certification program.

2.5 The Zigbee Device Objects (ZDO)

2.5.1 Scope

This section describes the concepts, structures, and primitives needed to implement a Zigbee Device Objects application on top of a Zigbee Application Support Sub-layer (section 2.2) and Zigbee Network Layer (Chapter 3).

Zigbee Device Objects are applications which employ network and application support layer primitives to implement Zigbee End Devices, Zigbee Routers, and Zigbee Coordinators.

The Zigbee Device Object Profile employs Clusters to describe its primitives. The Zigbee Device Profile Clusters do not employ attributes and are analogous to messages in a message transfer protocol. Cluster identifiers are employed within the Zigbee Device Profile to enumerate the messages employed within Zigbee Device Objects.

Zigbee Device Objects also employ configuration attributes. The configuration attributes within Zigbee Device Objects are attributes set by the application or stack profile. The configuration attributes are also not related to the Zigbee Device Profile, though both the configuration attributes and the Zigbee Device Profile are employed with Zigbee Device Objects.

2.5.2 Device Object Descriptions

The Zigbee Device Objects are an application solution residing within the Application Layer (APL) and above the Application Support Sub-layer (APS) in the Zigbee stack architecture as illustrated in Figure 1-1.

The Zigbee Device Objects are responsible for the following functions:

- Initializing the Application Support Sublayer (APS), Network Layer (NWK), Security Service Provider (SSP) and any other Zigbee device layer other than the end applications residing over Endpoints 1 – 254.
- Assembling configuration information from the end applications to determine and implement the functions described in the following sections.

2.5.2.1 Primary Discovery Cache Device Operation - Deprecated

2.5.2.2 Device and Service Discovery

This function SHALL support device and service discovery within a single PAN. Additionally, for all Zigbee device types, this function SHALL perform the following:

The following discovery features SHALL be supported:

Device Discovery:

- Based on a unicast inquiry of a Zigbee Coordinator or Zigbee Router’s IEEE address, the IEEE Address of the requested device plus, optionally, the NWK Addresses of all associated devices SHALL be returned.
- Based on a unicast inquiry of a Zigbee End Device’s IEEE address, the IEEE Address of the requested device SHALL be returned.
- Based on a broadcast inquiry (of any broadcast address type) of a Zigbee Coordinator or Zigbee Router’s NWK Address with a supplied IEEE Address, the NWK Address of the requested device plus, optionally, the NWK Addresses of all associated devices SHALL be returned.
- Based on a broadcast inquiry (of any broadcast address type) of a Zigbee End Device’s NWK Address with a supplied IEEE Address, the NWK Address of the requested device SHALL be returned. The responding device SHALL employ APS acknowledged service for the unicast response to the broadcast inquiry.

Service Discovery: Based on the following inputs, the corresponding responses SHALL be supplied:

- NWK address plus Active Endpoint query type – Specified device SHALL return the endpoint number of all applications residing in that device. Should the list of active endpoints exceed the ASDU size and where fragmentation is not supported on the server device, an extended version of the query type is also provided to return the full list through multiple requests.
- NWK address or broadcast address (of any broadcast address type) plus Service Match including Profile ID and, optionally, Input and Output Clusters – Specified device matches Profile ID with all active endpoints to determine a match. If no input or output clusters are specified, the endpoints that match the request are returned. If input and/or output clusters are provided in the request, those are matched as well, and any matches are provided in the response with the list of endpoints on the device providing the match. The responding device SHALL employ APS acknowledged service for the unicast response to the broadcast inquiry. By convention, in cases where the application profile enumerates input clusters and their response output clusters with the same cluster identifier, the application profile SHALL list only the input cluster within the Simple Descriptor for the purposes of Service Discovery.
- NWK address plus Node Descriptor or Power Descriptor query type – Specified device SHALL return the Node or Power Descriptor for the device.
- NWK address, Endpoint Number plus Simple Descriptor query type – Specified address SHALL return the Simple Descriptor associated with that Endpoint for the device. Should the list of input and/or output clusters exceed the ASDU size capacity to return the Simple Descriptor in a single packet an extended version of the query type is also provided to return the full list through multiple requests.

2.5.2.3 Security Manager

This function determines whether security is enabled or disabled and, if enabled, SHALL perform the following:

- Transport Key
- Request Key
- Update Device
- Remove Device
- Switch Key

The Security Manager function addresses the Security Services Specification (Chapter 4). The Security Management entity, implemented by APSME primitive calls by ZDO, performs the following:

- Transports the NWK Key from the Trust Center using secured communication with the Trust Center. This step employs the APSME-TRANSPORT-KEY primitive.
- Establishes or transports Link Keys, as required, with specific devices in the network. These steps employ the APSME-TRANSPORT-KEY and/or APSME-REQUEST-KEY primitives.
- Informs the Trust Center of any devices that join the network using the APSME-UPDATE-DEVICE primitives. This function is only performed if the device is a Zigbee router.
- Permits devices to obtain keys from the Trust Center using the APSME-REQUEST-KEY primitives.
- Permits the Trust Center to remove devices from the network using the APSME-REMOVE-DEVICE primitives.
- Permits the Trust Center to switch the active network key using the APSME-SWITCH-KEY primitives.

2.5.2.4 Network Manager

This function SHALL implement the Zigbee Coordinator, Zigbee Router, or Zigbee End Device logical device types according to configuration settings established either via a programmed application or during installation. If the device type is a Zigbee Router or Zigbee End Device, this function SHALL provide the ability to select an existing PAN to join and implement procedures which permit the device to rejoin if network communication is lost. If the device type is a Zigbee Coordinator or Zigbee Router, this function SHALL provide the ability to select an unused channel for creation of a new PAN. Note that it is possible to deploy a network without a device pre-designated as Zigbee Coordinator where the first Full Function Device (FFD) activated assumes the role of Zigbee Coordinator.

The following description covers processing addressed by Network Management:

- Permits specification of a channel list for network scan procedures. Default is to specify use of all channels in the selected band of operation.
- Manages network scan procedures to determine neighboring networks and the identity of their Zigbee coordinators and routers.
- Permits selection of a channel to start a PAN (Zigbee Coordinator) or selection of an existing PAN to join (Zigbee Router or Zigbee End Device).
- Supports orphaning and extended procedures to rejoin the network, including support for intra_PAN portability.
- May support direct join. For Zigbee Coordinators and Zigbee Routers, a local version of direct join MAY be supported to enable the device to join via the orphaning or rejoin procedures.
- May support Management Entities that permit external network management.
- Detects and reports interference to support changing network channels.
- Manages network interference reporting and selection of a new channel for network operation if interference exists on the initial channel if the particular node is identified as the network manager for the overall PAN.

2.5.2.5 Binding Manager

The Binding Manager performs the following:

- Establishes resource size for the Binding Table. The size of this resource is determined via a programmed application or via a configuration attribute defined during installation.
- Processes bind requests for adding or deleting entries from the APS binding table.
- Supports Bind and Unbind commands from external applications such as those that MAY be hosted on a commissioning or network management tool to support assisted binding. Bind and Unbind commands SHALL be supported via the Zigbee Device Profile (see section 2.4).
- Permits configuration tools to exchange one device for another in all the binding table entries which refer to it.

2.5.2.6 Node Manager

For Zigbee Coordinators and Zigbee Routers, the Node Management function performs the following:

- Permits remote management commands to perform network discovery.
- Provides remote management commands to retrieve the routing table.
- Provides remote management commands to retrieve the binding table.
- Provides a remote management command to have the device leave the network or to direct that another device leave the network.
- Provides a remote management command to retrieve the LQI for neighbors of the remote device.
- Provides a remote management command to Permit or disallow joining on particular routers or to generally allow or disallow joining via the Trust Center.

2.5.2.7 Group Manager

The Group Manager performs the following:

- Provides for inclusion of application objects within the local device into groups under application control.
- Provides for removal of application objects within the local device from group membership under application control.

2.5.3 Layer Interface Description

Unlike other device descriptors for applications residing above Endpoints 1 – 254, the Zigbee Device Objects (ZDO) interface to the APS via the APSME-SAP in addition to the APSDE-SAP. ZDO communicates over Endpoint 0 using the APSDE-SAP via Profiles like all other applications. The Profile used by ZDO is the Zigbee Device Profile (see section 2.4). ZDO frames SHALL NOT be fragmented.

Zigbee Device Objects SHALL employ Endpoint 0 as the source and destination endpoint in any transmitted Zigbee Device Profile request frames, and SHALL expect Endpoint 0 as the source and destination endpoint in any received response frames.

2.5.4 System Usage

2.5.4.1 Object Overview

Zigbee Device Objects contain six Objects:

- Device and Service Discovery
- Network Manager

- Binding Manager
- Security Manager
- Node Manager
- Group Manager

Table 2-130 describes these Zigbee Device Objects.

Table 2-130. Zigbee Device Objects

Object		Description
Name	**Status**	
:Device_and_Service_Discovery	M	Handles device and service discovery.
:Network_Manager	M	Handles network activities such as network discovery, leaving/joining a network, resetting a network connection and creating a network.
:Binding_Manager	O	Handles binding and unbinding activities.
:Security_Manager	M	Handles security services such as key loading, key establishment, key transport and authentication.
:Node_Manager	O	Handles management functions.
:Group Manager	O	Handles management of groups.

2.5.4.2 Optional and Mandatory Objects and Attributes

Objects listed as Mandatory SHALL be present on all Zigbee devices. However, for certain Zigbee logical types, Objects listed as Optional for all Zigbee devices MAY be Mandatory in specific logical device types. For example, the NLME-NETWORK-FORMATION.request within the Network_Manager object is in a Mandatory object and is an Optional attribute, though the attribute is required for Zigbee Coordinator logical device types. The introduction section of each Device Object section will detail the support requirements for Objects and Attributes by logical device type.

2.5.4.3 Security Key Usage

Zigbee Device Objects MAY employ security for packets created by Zigbee Device Profile primitives. These application packets using APSDE on Endpoint 0 SHALL utilize the APSDE Security Service Provider interface like all other Application Objects.

2.5.4.4 Public and Private Methods

Methods that are accessible to any endpoint application on the device are called public methods. Private methods are only accessible to the Device Application on endpoint 0 and not to the end applications (which run on endpoints 1 –254).

2.5.4.5 State Machine Functional Descriptions

2.5.4.5.1 Zigbee Coordinator

2.5.4.5.1.1 Initialization

The implementation SHALL set the startup-related IB attributes shown in Table 2-131 to values that reflect the desired startup behavior for the device. In particular, the *apsDesignatedCordinator* attribute of the IB SHALL be set to TRUE. If the device implements more than one option for Zigbee protocol version or stack profile, it SHALL choose a single value for each and set *nwkcProtocolVersion* and *nwkStackProfile* accordingly. Additionally, provision SHALL be made to provide configuration elements to describe the Node Descriptor, Power Descriptor, Simple Descriptor for each active endpoint and application plus the list of active endpoints. These configurations SHALL be embodied in :Config_Node_Descriptor, :Config_Power_Descriptor, and :Config_Simple_Descriptors.

If supported, provision SHALL be made to supply configuration elements for the maximum number of bind entries. These elements SHALL be embodied in:Config_Max_Bind.

To start as a Zigbee coordinator, the device application SHALL execute the startup procedure described in section 2.5.4.5.6.2 with startup attributes set as described above. This SHOULD have the effect of executing the procedure for network formation described in section 3.6.1.1. The device application SHALL set the *nwkSecurityLevel* and *nwkAllFresh* NIB attributes according to the values established by convention within the Stack Profile employed by the device. The device application SHALL check the return status via the NLME-NETWORK-FORMATION.confirm to verify successful creation of the PAN. The :Config_Permit_Join_Duration SHALL be set according to the default attribute value supplied using the NLME-PERMIT-JOINING.request. Additionally, the nwkNetworkBroadcastDeliveryTime and nwkcTransactionPersistenceTime Network Information Block attributes (see section 3.6.2) SHALL be set with :Config_NWK_BroadcastDeliveryTime and :Config_NWK_TransactionPersistenceTime respectively (see section 2.5.5).

Provision SHALL be made to ensure APS primitive calls from the end applications over EP 1 through EP 254 return appropriate error status values prior to completion of the Initialization state by Zigbee Device Objects and transition to the normal operating state.

2.5.4.5.1.2 Normal Operating State

In this state, the Zigbee Coordinator SHALL process the list of direct joined addresses in :Config_NWK_Join_Direct_Addrs by issuing an NLME-ADD-NEIGHBOR.request for each included address in the list. Processing of the direct joined addresses SHALL employ the :Config_Max_Assoc attribute in evaluating whether to successfully process a direct joined address within :Config_NWK_Join_Direct_Addrs.

The Zigbee coordinator SHALL allow other devices to join the network based on the configuration items :Config_Permit_Join_Duration and :Config_Max_Assoc. When a new device joins the network, the device application shall be informed via the NLME-JOIN.indication. Should the device be admitted to the PAN, the Zigbee coordinator SHALL indicate this via the NLME-JOIN.confirm with SUCCESS status.

The Zigbee coordinator SHALL respond to any device discovery or service discovery operations requested of its own device, The device application SHALL also ensure that the number of binding entries does not exceed the :Config_Max_Bind attribute.

The Zigbee coordinator SHALL support the NLME-PERMIT-JOINING.request and NLME-PERMIT-JOINING.confirm to permit application control of network join processing.

The Zigbee coordinator SHALL maintain a list of currently associated devices and facilitate support of orphan scan and rejoin processing to enable previously associated devices to rejoin the network. The Zigbee coordinator MAY support the ability for devices to be directly included in the network via the NLME-ADD-NEIGHBOR.request and NLME-ADD-NEIGHBOR.confirm. This feature SHALL permit lists of Zigbee IEEE addresses to be provided to the Zigbee coordinator and for those addresses to be included as previously associated devices. It SHALL be possible for Zigbee devices with those addresses to directly join the network via orphaning or rejoin procedures rather than associating directly.

The Zigbee coordinator SHALL support the NLME-NWK-STATUS.indication and process those notifications per section 3.2.2.31.

The Zigbee coordinator SHALL process Device_annce messages from other Zigbee devices. Upon receipt of a Device_annce, the Zigbee coordinator SHALL check all internal tables holding 64-bit IEEE addresses for devices within the PAN for a match with the address supplied in the Device_annce message. If a match is detected, the Zigbee coordinator SHALL update its *nwkAddressMap* attribute of the NIB corresponding to the matched 64- bit IEEE address to reflect the updated 16-bit NWK address contained in the Device_annce. The Zigbee Coordinator SHALL also employ the address conflict resolution procedure detailed in section 3.6.1.10.

2.5.4.5.1.3 Trust Center Operation

The network device pointed to by the address in *apsTrustCenterAddress* SHALL function as the Trust Center when security is enabled on the network.

The Trust Center operation is defined within section 4.6.2.

2.5.4.5.2 Zigbee Router

2.5.4.5.2.1 Initialization

The implementation SHALL set the startup-related IB attributes shown in Table 2-131 to values that reflect the desired startup behavior for the device. In particular, the *apsDesignatedCordinator* attribute of the IB SHALL be set to FALSE.

If supported, provision SHALL be made to supply configuration elements for the maximum number of bind entries. These elements SHALL be embodied in :Config_Max_Bind.

To start as a Zigbee router, the device application SHALL execute the startup procedure described in section 2.5.4.5.6.2 with startup attributes set as described above. This SHOULD have the effect of executing either the procedure for network rejoin or else the full procedure for network join through MAC association described in section 3.6.1.6.1. The NLME-NETWORK-AND-PARENT-DISCOVERY.request procedure SHALL be implemented :Config_NWK_Scan_Attempts, each separated in time by :Config_NWK_Time_btwn_Scans. The purpose of repeating the NLME-NETWORK-AND-PARENT-DISCOVERY.request is to provide a more accurate neighbor list and associated link quality indications to the NWK layer. Specification of the algorithm for selection of the PAN SHALL be left to the profile description and MAY include use of the Extended PAN ID, operational mode of the network, identity of the Zigbee Router or Coordinator identified on the PAN, depth of the Zigbee Router on the PAN from the Zigbee Coordinator for the PAN, capacity of the Zigbee Router or Coordinator, the routing cost, or the Protocol Version Number (these parameters are supplied by the NLME-NETWORK-AND-PARENT-DISCOVERY.confirm and the beacon payload).

The Zigbee router MAY join networks employing the current protocol version number or MAY join networks employing a previous protocol version number, under application control, if backward compatibility is supported in the device. A single Zigbee PAN SHALL consist of devices employing only a single protocol version number (networks with devices employing different protocol version numbers and frame formats within the same PAN are not permitted). An optional configuration attribute, :Config_NWK_alt_protocol_version, provides the protocol version numbers which the device MAY choose to employ other than the current protocol version number. Once the Zigbee router chooses a PAN and a specific protocol version number, it SHALL employ that protocol version number as its *nwkcProtocolVersion*. Additionally, the Zigbee router SHALL then adhere to all frame formats and processing rules supplied by the version of the Zigbee Specification employing that protocol version number.

The :Config_Permit_Join_Duration shall be set according to the default parameter value supplied using NLME-PERMIT-JOINING.request. The router SHALL support the NLME-START-ROUTER.request and NLME-START-ROUTER.confirm to begin operations as a router within the PAN it has joined. Additionally, the *nwkNetworkBroadcastDeliveryTime* and *nwkcTransactionPersistenceTime* Network Information Block attributes (see section 3.5.2) SHALL be set with :Config_NWK_BroadcastDeliveryTime and :Config_NWK_TransactionPersistenceTime respectively (see section 2.5.5).

Provision SHALL be made to ensure APS primitive calls from the end applications over EP 1 through EP 254 return appropriate error status values prior to completion of the Initialization state by Zigbee Device Objects and transition to the normal operating state.

If the network has security enabled, the device SHALL wait for successful acquisition of the NWK key to start functioning as a router in the network. See section 4.6.2 for details on Trust Center operations.

The device application SHALL set the *nwkSecurityLevel* NIB attribute to the values used in the network and begin functioning as a router using NLME-START-ROUTER.request.

2.5.4.5.2.2 Normal Operating State

In this state, the Zigbee router SHALL allow other devices to join the network based on the configuration items :Config_Permit_Join_Duration and :Config_Max_Assoc. When a new device joins the network, the device application SHALL be informed via the NLME-JOIN.indication attribute. Should the device be admitted to the PAN, the Zigbee router SHALL indicate this via the NLME-JOIN.confirm with SUCCESS status. If security is enabled on the network, the device application SHALL inform the Trust Center via the APSME-UPDATE-DEVICE. request.

The Zigbee router SHALL respond to any device discovery or service discovery operations requested of its own device. The device application SHALL also ensure that the number of binding entries does not exceed the :Config_Max_Bind attribute.

The Zigbee router SHALL support APSME-TRANSPORT-KEY.indication to receive keys from the Trust Center.

The Zigbee router SHALL support the NLME-PERMIT-JOINING.request and NLME-PERMIT-JOINING.confirm to permit application control of network join processing.

The Zigbee router SHALL support the NLME-NWK-STATUS.indication and process those notifications per section 3.2.2.31.

The Zigbee router SHALL support the NLME-LEAVE.request and NLME-LEAVE.confirm employing the :Config_NWK_Leave_removeChildren attribute where appropriate to permit removal of associated devices under application control. Conditions that lead to removal of associated devices MAY include lack of security credentials, removal of the device via a privileged application or detection of exception.

The Zigbee router SHALL process Device_annce messages from other Zigbee devices. Upon receipt of a Device_annce, the Zigbee router SHALL check all internal tables holding 64-bit IEEE addresses for devices within the PAN for a match with the address supplied in the Device_annce message. If a match is detected, the Zigbee router SHALL update its *nwkAddressMap* of the NIB corresponding to the matched 64-bit IEEE address to reflect the updated 16-bit NWK address contained in the Device_annce. The Zigbee Router SHALL also employ the address conflict resolution procedure detailed in section 3.6.1.10.

The Zigbee router SHALL maintain a list of currently associated end devices and facilitate support of orphan scan and rejoin processing to enable previously associated end devices to rejoin the network.

The Zigbee router MAY decide it has lost contact with the network it was joined to. In this situation, the router SHOULD conduct an active scan to find the network. If the network is found more than once the router SHOULD attempt to rejoin where there is a more recent value of *nwkUpdateId* in the beacon payload.

2.5.4.5.3 Binding Table Cache Operation – DEPRECATED

2.5.4.5.4 Operations to Support Intra-PAN Portability

2.5.4.5.4.1 Overview

The operations described in this section are carried out by Zigbee Coordinator and Zigbee Router Devices for support of intra-PAN portability.

The main steps are summarized as follows:

1. Detect the problem - The ZDO of the device is notified of acknowledgement failures via the NLME-NWK-STATUS.indication primitive, and identifies a problem. For end devices this can be initially detected by failures with their parent device. For routers, an application layer keepalive mechanism is required to determine that connectivity to important devices is lost, such as the Trust Center for centralized networks. The keepalive definition and failure detection is left up to the application layer and is outside the scope this document.
2. Carry out the NWK layer rejoin procedure - The ZDO of a ZED initiates this process using the NLME-JOIN.request primitive, either through a secured or un-secured rejoining procedure. The NWK rejoin procedures closely mirror the MAC association procedure.

3. Security verification - Secured and unsecured protocol steps are described to ensure that the orphaned device SHOULD really be accepted.

4. Inform the rest of the network - when a device changes parents the steps to complete address conflict detection in section 3.6.1.10 SHALL be completed. These actions also serve to notify the old parent that the End Device has changed parents.

5. Provide a means for parents that were temporarily unavailable and caused the end-device to rejoin are able to update their child tables once they are back online.

These steps are described in detail in the subsections below. The mechanism for secured rejoin of a ZED and for trust center rejoin of a ZED or ZR are both illustrated in Figure 2-91. Note that the NWK and SEC sections on secured and trust center rejoin (sections 3.2.2.13, 3.2.2.14, 3.2.2.15, 3.6.1.6, and 4.6.3) SHALL be the authoritative text for these procedures. The diagrams in this section are provided for illustrative purposes only.

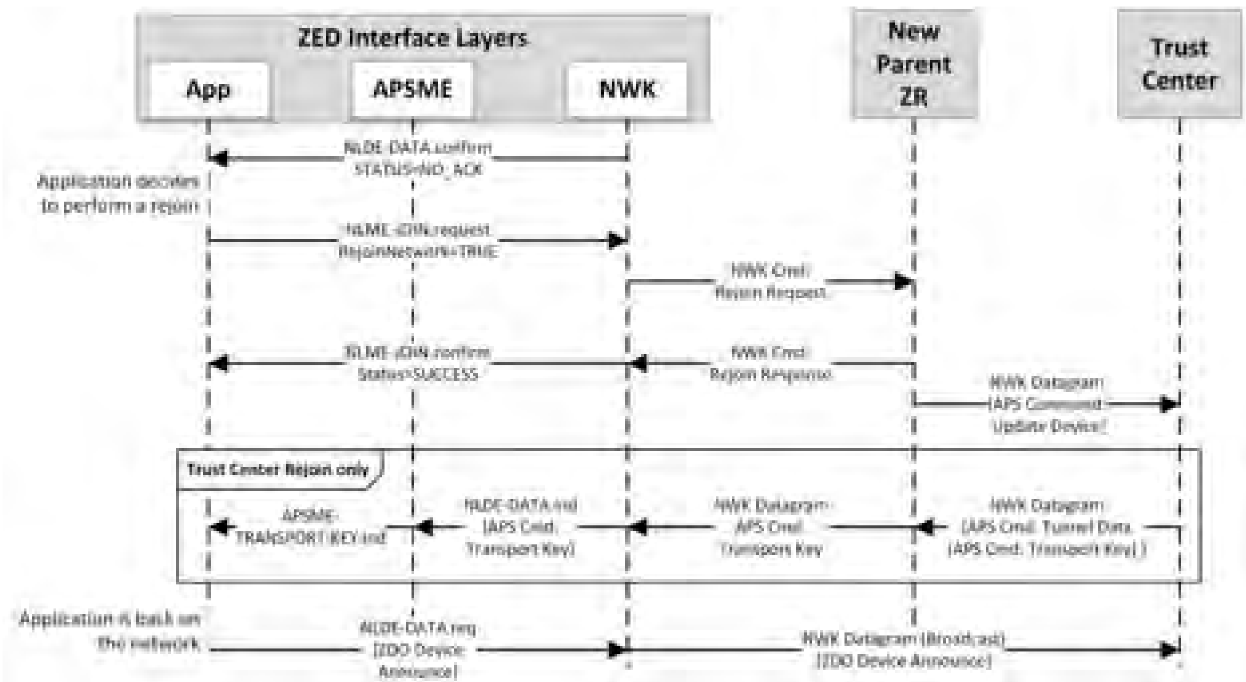


Figure 2-91. Portability Message Sequence Chart: ZED Rejoin

2.5.4.5.4.2 Description of Operations for Security Verification

As for MAC association, a Zigbee Coordinator or Zigbee Router device is informed of a rejoined device when the NLME issues an NLME-JOIN.indication primitive. This SHALL be handled in the same way as for an association indication, except that for a secured rejoin the update device and key transport step.

Full network operation SHALL NOT be permitted until the verification steps described below have been carried out.

Measures SHALL be taken by a newly (re-)joined node and by its new parent to verify that it is really allowed to be on this network. Two cases are envisioned:

One or the other is not implemented according to this specification, and SHOULD NOT have joined. The measures described here allow both sides to revoke the join in this case.

One or the other device is a compromised/hacked device. In the case that security is enabled, the measures in section 4.6.3.6 are additionally applied so that an unauthorized join is revoked.

This verification is carried out using existing commands. Section 2.5.4.5.4.3 describes the transmission of a Device_annce command to the new parent. The new parent SHALL check that this or some other message is correctly formed and contains the addressing fields corresponding to the orphaned device. If security is enabled, then this command SHALL be secured with the network key, and the new parent SHALL verify that all security processing is carried out correctly. If all these checks succeed then the orphaned device SHALL become joined to the network. Otherwise, it SHALL NOT become joined to the network at this time. As normal, messages sent from a device not joined to the network SHALL NOT be forwarded across the network, and commands SHALL NOT be carried out. Accordingly, the orphaned device SHALL only become joined to the network once it receives at least one correctly formed Zigbee message from the new parent. If security is enabled, this message SHALL be secured with the network key and all security processing SHALL be carried out correctly. If messages cannot be exchanged in protocol, then the orphaned device SHALL NOT become joined to the network at this time.

2.5.4.5.4.3 Description of Operations for Informing the Rest of the Network

If the Zigbee End Device rejoins a new parent using the orphaning of rejoin process it SHALL complete the address conflict process in section 3.6.1.9. Upon receiving the Device_annce, all devices SHALL check their internal tables holding 64-bit IEEE addresses for devices within the PAN for a match with the address supplied in the Device_annce message. If a match is detected, the device SHALL update the *nwkAddressMap* attribute of the NIB corresponding to the matched 64-bit IEEE address to reflect the updated 16-bit NWK address contained in the Device_annce. Devices will generally keep their existing NWK addresses during the intra-PAN portability procedure. Also, if the NWK address has changed during the intra-PAN portability procedure, the ZDO SHALL arrange that any IEEE address to short address mappings which have become known to applications running on this device be updated. This behavior is mandatory, but the mechanism by which it is achieved is outside the scope of this specification.

2.5.4.5.5 Zigbee End Device

2.5.4.5.5.1 Initialization

The implementation SHALL set the startup-related IB attributes shown in Table 2-131 to values that reflect the desired startup behavior for the device. In particular, the *apsDesignatedCordinator* attribute of the IB SHALL be set to FALSE.

If supported, provision SHALL be made to supply configuration elements for the maximum number of bind entries. These elements SHALL be embodied in :Config_Max_Bind.

To start as a Zigbee end device, the device application SHALL execute the startup procedure described in section 2.5.4.5.6.2 with startup parameters set as described above. This SHOULD have the effect of executing either the network rejoin or else the full procedure for network join through MAC association described in section 3.6.1.6.1. The NLME-NETWORK-AND-PARENT-DISCOVERY.request procedure shall be implemented :Config_NWK_Scan_Attempts, each separated in time by :Config_NWK_Time_btwn_Scans. The purpose of repeating the NLME-NETWORK-AND-PARENT-DISCOVERY.request is to provide a more accurate neighbor list and associated link quality indications to the NWK layer. Specification of the algorithm for selection of the PAN SHALL be left to the profile description and MAY include use of the Extended PAN ID, operational mode of the network, identity of the Zigbee Router or Coordinator identified on the PAN, depth of the Zigbee Router on the PAN from the Zigbee Coordinator for the PAN, capacity of the Zigbee Router or Coordinator, the routing cost, or the Protocol Version Number (these parameters are supplied by the NLME-NETWORK-AND-PARENT-DISCOVERY.confirm and the beacon payload).

The Zigbee end device MAY join networks employing the current protocol version number or MAY join networks employing a previous protocol version number, under application control, if backward compatibility is supported in the device. A single Zigbee PAN SHALL consist of devices employing only a single protocol version number (networks with devices employing different protocol version numbers and frame formats within the same PAN are not permitted). An optional configuration attribute, :Config_NWK_alt_protocol_version, provides the protocol version numbers which the device MAY choose to employ other than the current protocol version number. Once the Zigbee end device chooses a PAN and a specific protocol version number, it SHALL employ that protocol version number as its *nwkcProtocolVersion*. Additionally, the Zigbee end device SHALL then adhere to all frame formats and processing rules supplied by the version of the Zigbee Specification employing that protocol version number.

If the device application sets the NLME-JOIN RxOnWhenIdle parameter to FALSE, the :Config_NWK_indirectPollRate SHALL be used to determine the polling rate for indirect message requests. The :Config_NWK_indirectPollRate SHALL be set according to the value established by the application profile(s) supported on the device. Once polling for indirect message requests is initiated, if communications failure with the parent is detected determined by failure of indirect message requests :Config_Parent_Link_Threshold_Retry consecutive attempts, the device application SHALL employ the network rejoin procedure.

Once the End Device has successfully joined a network, the device SHALL issue a Device_annce providing its 64-bit IEEE address and 16-bit NWK address.

Provision SHALL be made to ensure APS primitive calls from the end applications over EP 1 through EP 254 return appropriate error status values prior to completion of the Initialization state by Zigbee Device Objects and transition to the normal operating state.

If network has security enabled, the device SHALL wait successful acquisition of the NWK key to start functioning as an end device in the network. See section 4.6.2 for details on Trust Center operations.

2.5.4.5.5.2 Normal Operating State

If the device application set the NLME-JOIN RxOnWhenIdle parameter to FALSE, the :Config_NWK_indirectPollRate SHALL be used to poll the parent for indirect transmissions while in the normal operating state. While a fragmented message is being received, the device MAY temporarily increase its polling rate, and SHALL ensure that it polls its parent at least once every macTransactionPersistenceTime seconds.

The Zigbee end device SHALL respond to any device discovery or service discovery operations requested of its own device using the attributes described in section 2.5.4.

The Zigbee end device SHALL support APSME-TRANSPORT-KEY.indication to receive keys from the Trust Center.

The Zigbee End Device SHALL process Device_annce messages from other Zigbee devices. Upon receipt of a Device_annce, the Zigbee End Device SHALL check all internal tables holding 64-bit IEEE addresses for devices within the PAN for a match with the address supplied in the Device_annce message. If a match is detected, the Zigbee End Device SHALL update the *nwkAddressMap* of the NIB corresponding to the matched 64-bit IEEE address to reflect the updated 16-bit NWK address contained in the Device_annce.

The Zigbee End Device SHALL process the NLME-NWK-STATUS.indication sent from the NWK layer. If the error code equals to 0x09 (Parent Link Failure), the ZED will update its failure counter maintained in ZDO. If the value of the failure counter is smaller than the :Config_Parent_Link_Retry_Threshold attribute, the ZED MAY decide to issue further commands to attempt to communicate with the parent node, depending on the application of the ZED. If the value of the failure counter exceeds the :Config_Parent_Link_Retry_Threshold attribute, the ZED SHALL then prepare to start the rejoin process. Note that implementers MAY optionally use a more accurate time-windowed scheme to identify a link failure.

The rejoin process mirrors the MAC association process very closely, however, a device is permitted to rejoin a parent that is not accepting new associations. The ZDO MAY use the NLME-NETWORK-AND-PARENT-DISCOVERY.request primitive to detect potential alternative parents, and in order to optimize recovery latency and reliability, SHALL select an appropriate new parent based on the following information from that device's beacon:

- PAN ID
- EPID (Extended PAN ID)
- Channel
- Signal strength
- Whether the potential parent indicates that it is currently able to communicate with its Trust Center
- Whether this device has recently failed to join this parent, or this network

Once a potential parent has been selected, the ZDO SHALL issue an NLME-JOIN.request primitive with RejoinNetwork set to 0x02.

The start time of the rejoin process is determined by the time the last NLME-JOIN.request primitive was sent and by the attribute :Config_Rejoin_Interval. Only if the interval between the current and the previous NLME-JOIN.request sent time is longer than the :Config_Rejoin_Interval SHALL a new NLME-JOIN.request primitive be sent. The application MAY want to gradually increase the :Config_Rejoin_Interval if a certain number of retries have been done (or a certain period of time has passed) but none of them were successful. The :Config_Rejoin_Interval SHOULD NOT exceed the :Config_Max_Rejoin_Interval. Every time an NLME-JOIN.confirm has been successfully received, the ZDO SHALL reset its failure counter to zero and the :Config_Rejoin_Interval attribute to its initial value. The choice of the default initial value and the algorithm of increasing the rejoin interval shall be determined by the application, and is out of the scope of this document.

If the Zigbee End Device rejoins a new parent using the rejoin process, it SHALL complete the address conflict process in section 3.6.1.10.

2.5.4.5.6 Support for Commissioning Applications

Zigbee devices in the field will need commissioning, and it will be up to developers to provide applications that perform such commissioning. There is a risk that applications from different vendors will work differently, thereby diminishing the ability of Zigbee devices from different vendors to operate seamlessly on the same network. As a partial solution to this problem, this section lists a common set of configuration attributes for Zigbee devices and outlines a common procedure for devices to use at start-up time. The other critical component of the solution is a common set of commissioning protocols and procedures, which are outside the scope of this document.

2.5.4.5.6.1 Configuration Attributes

The startup procedure outlined in section 2.5.4.5.6.2 is designed in such a way that, by using it consistently, devices can go through all the stages of commissioning up to being joined to the proper Zigbee network and able to send and receive application data traffic. Later-stage commissioning, including the commissioning of bindings and group membership is discussed briefly in section 2.5.4.5.6.3. The procedure makes use of the system attributes listed in Table 2-131.

Table 2-131. Startup Attributes

Name	Reference	Comment
nwkExtendedPANID	Table 3.43	This is the extended PANID of the network to which the device is joined. If it has a value of 0x0000000000000000, then the device is not connected to a network.
apsDesignatedCoordinator	Table 2-24	This Boolean flag indicates whether the device SHOULD assume on startup that it SHALL become a Zigbee coordinator.
apsChannelMaskList	Table 2-24	This is a list containing one or more masks which define the allowable channels on which the device MAY attempt to form or join a network at startup time.
apsUseExtendedPANID	Table 2-24	The 64-bit identifier of the network to join or form.
apsUseInsecureJoin	Table 2-24	A Boolean flag that indicates if it is OK to use insecure join on startup.

2.5.4.5.6.2 Startup Procedure

The startup procedure uses the attributes listed in section 2.5.4.5.6.1 to perform a controlled startup of the Zigbee networking facilities of a device. The procedure SHOULD be run whenever the device restarts, but MAY also be run under application control at the discretion of the developer.

When a device starts up, it SHOULD check the value of *nwkExtendedPANID*. If *nwkExtendedPANID* has a non-zero value, then the device SHOULD assume it has all the network parameters required to operate on a network. Note that the device SHOULD assume the channel identifier present in its current network parameters but MAY need to scan over the ChannelMask if the *nwkExtendedPANID* is not found. In order for this to work effectively across power failures and processor resets, *nwkExtendedPANID* SHALL be placed in non-volatile storage.

If the device has all necessary parameters to operate on a network, the application MAY choose to send a ZDO Device Announce. However, it is recommended that the stack jitter this message by apsParentAnnounceBaseTimer + random(apsParentAnnounceJitterMax) seconds after reset to avoid a network-wide power cycle triggering devices to flood the network with broadcasts

If the device finds it is not connected to a network, then it SHOULD check the value of *apsDesignatedCoordinator*. If this attribute has a value of TRUE, then the device SHOULD follow the procedures for starting a network outlined in section 3.6.1.6 and SHOULD use the value of *apsChannelMaskList* for the ScanChannelsListStructure parameter of the NLME-NETWORK-FORMATION.request primitive, and set nwkExtendedPANID to the value given in *apsUseExtendedPANID* if *apsUseExtendedPANID* has a non-zero value.

If the device is not the designated coordinator and *apsUseExtendedPANID* has a non-zero value, the device SHOULD attempt to verify connectivity to the network specified in apsUseExtendedPANID. Verifying connectivity may be done via an NLME-JOIN.request with RejoinNetwork=TRUE and ExtendedPanID equal to apsUseExtendedPANID, or an NLME-SYNC.request, or by sending a NWK Command End Device Timeout Request. If the device receives a correctly formatted response frame this indicates successful connectivity.[3]

If the network rejoin attempt fails, and the value of the *apsUseInsecureJoin* attribute of the AIB has a value of TRUE, then the device SHOULD follow the procedure outlined in section 3.6.1.6 for joining a network, using *apsChannelMaskList* any place that a ScanChannelsListStructure mask is called for. If *apsUseExtendedPANID* has a non-zero value, then the device SHOULD join only the specified network and the procedure SHOULD fail if that network is found to be inaccessible. If *apsUseExtendedPANID* is equal to 0x0000000000000000, then the device SHOULD join the best available network.

2.5.4.5.6.3 Further Commissioning

Once a device is on a network and capable of communicating with other devices on the network in a secure manner, other commissioning becomes possible. Other items that SHOULD be subject to commissioning are shown in Table 2-132.

Table 2-132. Additional Commissioning Attributes

Name	Reference	Comment
apsBindingTable	Table 2-24	The binding table for this device. Binding provides a separation of concerns in the sense that applications MAY operate without having to manage recipient address information for the frames they emit. This information can be input at commissioning time without the main application on the device even being aware of it.
nwkSecurityMaterialSet	Table 4-2	This set contains the network keying material, which SHOULD be accessible to commissioning applications.

[3] CCB 3346

Name	Reference	Comment
apsDeviceKeyPairSet	Table 4-35	This is the set of link key pairs for devices that it wants to communicate using application layer encryption.
apsTrustCenterAddress	Table 4-35	The IEEE address of the Trust Center.
nwkNetworkAddress	Table 3-62	Commissioning applications MAY set the network short address of devices as long as address conflicts that MAY arise as a result are subject to address conflict resolution as described in section 3.6.1.10.

2.5.4.6 Network Manager

The Network Management function supports:

- Network Discovery
- Network Formation
- Permit/Disable Associations
- Association and Disassociation
- Route Discovery
- Network Reset
- Radio Receiver State Enable/Disable
- Get and Set of Network Management Information Block Data
- Detecting and reporting interference
- Receive network interference reports and change network channels if the particular node is identified as the network manager for the overall PAN

Network Management performs the above functions with NLME-SAP primitives (see Chapter 3).

2.5.4.6.1 Optional and Mandatory Attributes Within Network Manager

The Network Manager is a mandatory object for all Zigbee Device Types.

The Network Discovery, Get, and Set attributes (both requests and confirms) are mandatory for all Zigbee logical device types.

If the Zigbee logical device type is Zigbee Coordinator, the NWK Formation request and confirm, the NWK Leave request, NWK Leave indication, NWK Leave confirm, NWK Join indication, NWK Permit Joining request, NWK Permit Joining confirm, NWK Route Discovery request, and NWK Route Discovery confirm SHALL be supported. The NWK Direct Join request and NWK Direct Join confirm MAY be supported. The NWK Join request and the NWK Join confirm SHALL NOT be supported.

If the Zigbee logical device type is Zigbee Router, the NWK Formation request and confirm SHALL NOT be supported except if forming distributed networks. Additionally, the NWK Start Router request, NWK Start Router confirm, NWK Join request, NWK Join confirm, NWK Join indication, NWK Leave request, NWK Leave confirm, NWK Leave indication, NWK Permit Joining request, NWK Permit Joining confirm, NWK Route Discovery request, and NWK Route Discovery confirm SHALL be supported. The NWK Direct Join request and NWK Direct Join confirm MAY be supported.

If the Zigbee logical device type is Zigbee End Device, the NWK Formation request and confirm plus the NWK Start Router request and confirm SHALL NOT be supported. Additionally, the NWK Join indication and NWK Permit Joining request SHALL NOT be supported. The NWK Join request, NWK Join confirm, NWK Leave request, NWK Leave indication, NWK Leave confirm SHALL be supported.

For all Zigbee logical devices types, the NWK Sync request, indication and confirm plus NWK reset request and confirm plus NWK route discovery request and confirm SHALL be optional. Table 2-133 summarizes Network Manager Attributes. See Chapter 3 for a description of any of the primitives listed in Table 2-133.

For all Zigbee logical device types, reception of the NWK Network Status indication SHALL be supported, but no action is required in this version of the specification.

Table 2-133. Network Manager Attributes

Attribute	M/O	Type
NLME-GET.request	M	Private
NLME-GET.confirm	M	Private
NLME-SET.request	M	Private
NLME-SET.confirm	M	Private
NLME-NETWORK-AND-PARENT-DISCOVERY.request	M	Public
NLME-NETWORK-AND-PARENT-DISCOVERY.confirm	M	Public
NLME-NETWORK-FORMATION.request	O	Private
NLME-NETWORK-FORMATION.confirm	O	Private
NLME-START-ROUTER.request	O	Private
NLME-START-ROUTER.confirm	O	Private
NLME-JOIN.request	O	Private
NLME-JOIN.confirm	O	Private
NLME-JOIN.indication	O	Private
NLME-PERMIT-JOINING.request	O	Public
NLME-PERMIT-JOINING.confirm	O	Public
NLME-ADD-NEIGHBOR.request	O	Public
NLME-ADD-NEIGHBOR.confirm	O	Public
NLME_LEAVE.request	M	Public

Attribute	M/O	Type
NLME-LEAVE.confirm	M	Public
NLME_LEAVE.indication	M	Public
NLME-RESET.request	O	Private
NLME-RESET.confirm	O	Private
NLME-SYNC.request	O	Public
NLME-SYNC.indication	O	Public
NLME-SYNC.confirm	O	Public
NLME-NWK-STATUS.indication	M	Private
NLME-ROUTE-DISCOVERY.request	O	Public
NLME-ROUTE-DISCOVERY.confirm	O	Private
NLME-ED-SCAN.request	O	Private
NLME-ED-SCAN.confirm	O	Private
NLME-START-BACKOFF.request	O	Private

A single device in the network can become the Network Channel Manager. The operation of the network channel manager is described in Annex E. All other devices in the network are responsible for tracking message delivery failures and reporting interference in accordance with Annex E.

2.5.4.7 Node Manager

The Node Manager supports the ability to request and respond to management functions. These management functions only provide visibility to external devices regarding the operating state of the device receiving the request.

2.5.4.8 Group Manager

The Group Manager supports the ability to include application objects within groups or to remove application objects from groups. The group management functions operate only on application objects within the local device. Mechanisms to manage groups on other devices are beyond the scope of this document.

2.5.5 Configuration Attributes

This attribute is used to represent the minimum mandatory and/or optional attributes used as configuration attributes for a device summarized in Table 2-134.

Table 2-134. Configuration Attributes

Attribute	M/O	Type
:Config_Node_Descriptor	M	Public
:Config_Power_Descriptor	M	Public
:Config_Simple_Descriptors	M	Public
:Config_NWK_Scan_Attempts	M	Private
:Config_NWK_Time_btwn_Scans	M	Private
:Config_Complex_Descriptor	Deprecated	N/A
:Config_User_Descriptor	Deprecated	N/A
:Config_Max_Bind	O	Private
:Config_Permit_Join_Duration	O	Public
:Config_NWK_Security_Level	O	Private
:Config_NWK_Secure_All_Frames	O	Private
:Config_NWK_BroadcastDeliveryTime	O	Private
:Config_NWK_TransactionPersistenceTime	O	Private
:Config_NWK_indirectPollRate	O	Private
:Config_Max_Assoc	O	Private
:Config_NWK_Join_Direct_Addrs	O	Public
:Config_Parent_Link_Retry_Threshold	O	Public
:Config_Rejoin_Interval	O	Public
:Config_Max_Rejoin_Interval	O	Public

2.5.5.1 Configuration Attribute Definitions

Table 2-135. Configuration Attribute Definitions

Attribute	Description	When Updated
:Config_Node_Descriptor	Contents of the Node Descriptor for this device (see section 2.3.2.3).	The :Config_Node_Descriptor is either created when the application is first loaded or initialized with a commissioning tool prior to when the device begins operations in the network. It is used for service discovery to describe node features to external inquiring devices.
:Config_Power_Descriptor	Contents of the Power Descriptor for this device (see section 2.3.2.4).	The :Config_Power_Descriptor is either created when the application is first loaded or initialized with a commissioning tool prior to when the device begins operations in the network. It is used for service discovery to describe node power features to external inquiring devices.
:Config_Simple_Descriptors	Contents of the Simple Descriptor(s) for each active endpoint for this device (see section 2.3.2.5).	The :Config_Simple_Descriptors are created when the application is first loaded and are treated as “read-only.” The Simple Descriptor are used for service discovery to describe interfacing features to external inquiring devices.
:Config_NWK_Scan_Attempts	Integer value representing the number of scan attempts to make before the NWK layer decides which Zigbee coordinator or router to associate with (see section 2.5.4.5.1). This attribute has default value of 5 and valid values between 1 and 255.	The :Config_NWK_Scan_Attempts is employed within ZDO to call the NLME-NETWORK-AND-PARENT-DISCOVERY.request primitive the indicated number of times (for routers and end devices).
:Config_NWK_Time_btwn_Scans	Integer value representing the time duration (in OctetDurations) between each NWK discovery attempt described by :Config_NWK_Scan_Attempts (see section). This attribute has a default value of 0xc35 OctetDurations (100 milliseconds on 2.4GHz) and valid values between 1 and	The Config_NWK_Time_btwn_Scans is employed within ZDO to provide a time duration between the NLME-NETWORK-AND-PARENT-DISCOVERY.request attempts.

Attribute	Description	When Updated
	0x1f3fe1 OctetDurations (65535 milliseconds on 2.4GHz).	
:Config_Max_Bind	A constant which describes the maximum number of binding entries permitted.	The :Config_Max_Bind is a maximum number of supported Binding Table entries for this device.
:Config_Permit_Join_Duration	Permit Join Duration value set by the NLME-PERMIT-JOINING.request primitive (see Chapter 3).	The default value for :Config_Permit_Join_Duration is 0x00, however, this value can be established differently according to the needs of the profile.
:Config_NWK_Security_Level	Security level of the network (see Chapter 3).	This attribute is used only on the Trust Center and is used to set the level of security on the network.
:Config_NWK_Secure_All_Frames	If all network frames SHOULD be secured (see Chapter 3).	This attribute is used only on the Trust Center and is used to determine if network layer security SHALL be applied to all frames in the network.
:Config_NWK_BroadcastDeliveryTime	See Table 2-134.	The value for this configuration attribute is established in the Stack Profile.
:Config_NWK_TransactionPersistenceTime	See Table 2-134.. This attribute is mandatory for the Zigbee coordinator and Zigbee routers and not used for Zigbee End Devices.	The value for this configuration attribute is established in the Stack Profile.

Attribute	Description	When Updated
:Config_NWK_Alt_protocol_version	Sets the list of protocol version numbers, other than the current protocol version number, that the device MAY choose to employ in a PAN that it joins. This attribute is applicable only to Zigbee routers or end devices. The protocol version numbers in the list SHALL refer to older versions of the Zigbee Specification.	:Config_NWK_ Alt_protocol_version permits Zigbee routers and Zigbee end devices to join networks discovered that employ an earlier version of the Zigbee Specification; Since this attribute is optional, devices MAY also be created omitting this attribute which require only the current version of the Zigbee Specification; This attribute would be omitted in cases where certain features are required that are contained only in the current specification or where code size is limited in the device.
:Config_NWK_indirectPollRate	Sets the poll rate, in milliseconds, for the device to request indirect transmission messages from the parent.	The value for this configuration attribute is established by the application profile deployed on the device.
:Config_Max_Assoc	Sets the maximum allowed associations, either of routers, end devices, or both, to a parent router or coordinator.	The value for this configuration attribute is established by the stack profile in use on the device. Note that for some stack profiles, the maximum associations MAY have a dimension which provides for separate maximums for router associations and end device associations.
:Config_NWK_Join_Direct_Addrs	Consists of the following fields: • DeviceAddress - 64-bit IEEE address for the device to be direct joined. • CapabilityInformation - Operating capabilities of the device to be direct joined. • Link Key- If security is enabled, link key for use in the key-pair descriptor for this new device (see Table 4-36). See section 3.2.2.16 for details.	:Config_NWK_Join_Direct_Addrs permits the Zigbee Coordinator or Router to be pre-configured with a list of addresses to be direct joined.

Attribute	Description	When Updated
:Config_Parent_Link_Retry_Threshold	Contents of the link retry threshold for parent link (see section 2.5.4.5.5.2).	The Config_Parent_Link_Retry_Threshold is either created when the application is first loaded or initialized with a commissioning tool. It is used for the ZED to decide how many times it SHOULD retry to connect to the parent router before initiating the rejoin process.
:Config_Rejoin_Interval	Contents of the rejoin interval (see section 2.5.4.5.5.2).	The :Config_Rejoin_Interval is either created when the application is first loaded or initialized with a commissioning tool. It is used by the ZED to decide how often it SHOULD initiate the rejoin process.
:Config_MAX_Rejoin_Interval	Contents of the maximal rejoin interval (see section 2.5.4.5.5.2).	The :Config_MAX_Rejoin_Interval is either created when the application is first loaded or initialized with a commissioning tool. It is used by the ZED to set the maximum value permitted for :Config_Rejoin_Interval during the rejoin procedure.

CHAPTER 3. NETWORK SPECIFICATION

3.1 General Description

3.1.1 Network (NWK) Layer Overview

The network layer is required to provide functionality to ensure correct operation of the IEEE Std 802.15.4 MAC sub-layer and to provide a suitable service interface to the application layer. To interface with the application layer, the network layer conceptually includes two service entities that provide the necessary functionality. These service entities are the data service and the management service. The NWK layer data entity (NLDE) provides the data transmission service via its associated SAP, the NLDE-SAP, and the NWK layer management entity (NLME) provides the management service via its associated SAP, the NLME-SAP. The NLME utilizes the NLDE to achieve some of its management tasks and it also maintains a database of managed objects known as the network information base (NIB).

3.1.2 Network Layer Data Entity (NLDE)

The NLDE SHALL provide a data service to allow an application to transport application protocol data units (APDU) between two or more devices. The devices themselves SHALL be located on the same network.

The NLDE will provide the following services:

- **Generation of the Network level PDU (NPDU):** The NLDE SHALL be capable of generating an NPDU from an application support sub-layer PDU through the addition of an appropriate protocol header.
- **Topology-specific routing:** The NLDE SHALL be able to transmit an NPDU to an appropriate device that is either the final destination of the communication or the next step toward the final destination in the communication chain.
- **Security:** The ability to ensure both the authenticity and confidentiality of a transmission.

3.1.2.1 Network Layer Management Entity (NLME)

The NLME SHALL provide a management service to allow an application to interact with the stack.

The NLME SHALL provide the following services:

- **Configuring a new device:** this is the ability to sufficiently configure the stack for operation as required. Configuration options include beginning an operation as a Zigbee coordinator or joining an existing network.
- **Starting a network:** this is the ability to establish a new network.
- **Joining, rejoining and leaving a network:** this is the ability to join, rejoin or leave a network as well as the ability of a Zigbee coordinator or Zigbee router to request that a device leave the network.
- **Addressing:** this is the ability of Zigbee coordinators and routers to assign addresses to devices joining the network.
- **Neighbor discovery:** this is the ability to discover, record, and report information pertaining to the one-hop neighbors of a device.
- **Route discovery:** this is the ability to discover and record paths through the network, whereby messages MAY be efficiently routed.
- **Reception control:** this is the ability for a device to control when the receiver is activated and for how long, enabling MAC sub-layer synchronization or direct reception.
- **Routing:** this is the ability to use different routing mechanisms such as unicast, broadcast, many-to-one, or source routing to efficiently exchange data in the network.

3.2 Service Specification

Figure 3-1 depicts the components and interfaces of the NWK layer.

The NWK layer provides two services, accessed through two service access points (SAPs). These are the NWK data service, accessed through the NWK layer data entity SAP (NLDE-SAP), and the NWK management service, accessed through the NWK layer management entity SAP (NLME-SAP). These two services provide the interface between the application and the MAC sub-layer, via the MCPS-SAP and MLME-SAP interfaces (see [B1]). In addition to these external interfaces, there is also an implicit interface between the NLME and the NLDE that allows the NLME to use the NWK data service.

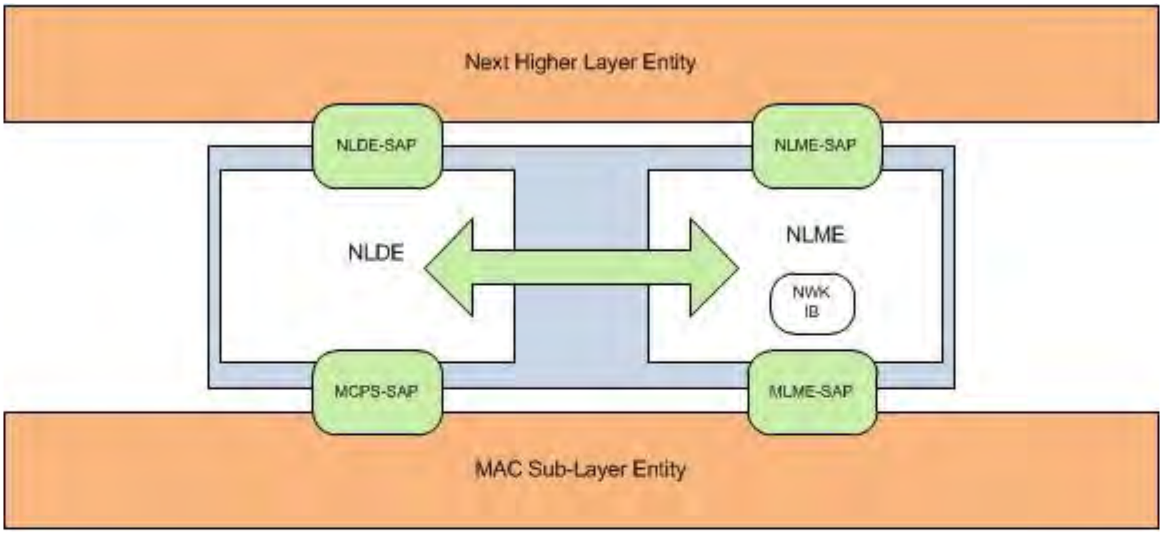


Figure 3-1. The NWK Layer Reference Model

3.2.1 NWK Data Service

The NWK layer data entity SAP (NLDE-SAP) supports the transport of application protocol data units (APDUs) between peer application entities. Table 3-1 lists the primitives supported by the NLDE-SAP and the sections in which these primitives are discussed.

Table 3-1. NLDE-SAP Primitives

NLDE-SAP Primitive	Request	Confirm	Indication
NLDE-DATA	3.2.1.1	3.2.1.2	3.2.1.3

3.2.1.1 NLDE-DATA.request

This primitive requests the transfer of a data PDU (NSDU) from the local APS sub-layer entity to a single or multiple peer APS sub-layer entities.

3.2.1.1.1 Semantics of the Service Primitive

The semantics of this primitive are as follows:

```
NLDE-DATA.request        {
                         DstAddrMode,
                         DstAddr,
                         NsduLength,
                         Nsdu,
                         NsduHandle,
                         UseAlias,
                         AliasSrcAddr,
                         AliasSeqNumber,
                         Radius,
                         DiscoverRoute,
                         SecurityEnable
                         }
```

Table 3-2 specifies the parameters for the NLDE-DATA.request primitive. Support of the additional parameters UseAlias, AliasSrcAddr, AliasSeqNumb in the NLDE-DATA.request primitive is required if GP feature is to be supported by the implementation.

Table 3-2. NLDE-DATA.request Parameters

Name	**Type**	**Valid Range**	**Description**
DstAddrMode	Integer	0x02	The type of destination address supplied by the DstAddr parameter. 0x02=16-bit network address of a device or a 16-bit broadcast address
DstAddr	16-bit address	0x0000 – 0xffff	Destination address.
NsduLength	Integer	*0 to aMaxPHYPacketSize - (nwkcMACFrameOverhead + nwkcMinHeaderOverhead)*	The number of octets comprising the NSDU to be transferred.
Nsdu	Set of octets	-	The set of octets comprising the NSDU to be transferred.
NsduHandle	Integer	0x00 – 0xff	The handle associated with the NSDU to be transmitted by the NWK layer entity.

Name	Type	Valid Range	Description
UseAlias	Boolean	TRUE or FALSE	The next higher layer MAY use the UseAlias parameter to request alias usage by NWK layer for the current frame. If the *UseAlias* parameter has a value of FALSE, meaning no alias usage, then the parameters *AliasSrcAddr* and *AliasSeqNumb* will be ignored. Otherwise, a value of TRUE denotes that the values supplied in *AliasSrcAddr* and *AliasSeqNumb* are to be used.
AliasSrcAddr	16-bit address	Any valid device address except a broadcast address	The source address to be used for this NSDU. If the *UseAlias* parameter has a value of FALSE, the AliasSrcAddr parameter is ignored.
AliasSeqNumb	Integer	0x00 – 0xff	The sequence number to be used for this NSDU. If the *UseAlias* parameter has a value of FALSE, the *AliasSeqNumb* parameter is ignored.
Radius	Unsigned integer	0x00 – 0xff	The distance, in hops, that a frame will be allowed to travel through the network.
DiscoverRoute	Integer	0x00 – 0x01	The DiscoverRoute parameter MAY be used to control route discovery operations for the transit of this frame (see section 3.6.4.5): 0x00 = suppress route discovery 0x01 = enable route discovery
SecurityEnable	Boolean	TRUE or FALSE	The SecurityEnable parameter MAY be used to enable NWK layer security processing for the current frame. If the *nwkSecurityLevel* attribute of the NIB has a value of 0, meaning no security, then this parameter will be ignored. Otherwise, a value of TRUE denotes that the security processing specified by the security level will be applied, and a value of FALSE denotes that no security processing will be applied.

3.2.1.1.2 When Generated

This primitive is generated by a local APS sub-layer entity whenever a data PDU (NSDU) is to be transferred to a peer APS sub-layer entity.

3.2.1.1.3 Effect on Receipt

If this primitive is received on a device that is not currently associated, the NWK layer will issue an NLDE-DATA.confirm primitive with a status of INV_REQUESTTYPE.

On receipt of this primitive, the NLDE first constructs an NPDU in order to transmit the supplied NSDU. If, during processing, the NLDE issues the NLDE-DATA.confirm primitive prior to transmission of the NSDU, all further processing is aborted. In constructing the new NPDU, the destination address field of the NWK header will be set to the value provided in the DstAddr parameter. If the UseAlias parameter has a value of TRUE, the source address field of the NWK header of the frame will be set to the value provided in the AliasSrcAddr parameter. If the UseAlias parameter has a value of FALSE, then the source address field will have the value of the *macShortAddress* attribute in the MAC PIB. The discover route sub-field of the frame control field of the NWK header will be set to the value provided in the DiscoverRoute parameter. If the supplied Radius parameter does not have a value of zero, then the radius field of the NWK header will be set to the value of the Radius parameter. If the Radius parameter has a value of zero, then the radius field of the NWK header will be set to twice the value of the *nwkcMaxDepth* attribute of the NIB. If the UseAlias parameter has a value of TRUE, the sequence number field of the NWK header of the frame will be set to the value provided in the AliasSeqNumb parameter. If the UseAlias parameter has a value of FALSE, then the NWK layer will generate a sequence number for the frame as described in section 3.6.2.1 and the sequence number field of the NWK header of the frame will be set to this sequence number value.

Once the NPDU is constructed, the NSDU is routed using the procedure described in section 3.6.4.3 if it is a unicast, or section 3.6.5 if it is a broadcast. When the routing procedure specifies that the NSDU is to be transmitted, this is accomplished as follows.

1. The device SHALL find the next hop address determined by the addressing mode and the routing procedure.
2. If the next hop is the MAC broadcast address 0xFFFF, do the following:
   a) Determine the set of currently active MAC interfaces, comprising all entries in the nwkMacInterfaceTable, where the Enabled field is set as TRUE.
   b) If this set is empty, proceed as follows:
      i) Issue an NLDE-DATA.confirm primitive with a status value of INVALID_INTERFACE.
      ii) No further processing SHALL be done.
   c) If the set is not empty, go to step 6 and repeat for each MAC SAP instance in the set of active interfaces.
3. Examine the nwkNeighborTable and find the entry where the Network Address element is equal to the next hop.
4. If no entry can be found, the NLDE SHALL do the following.
   a. Issue an NLDE-DATA.confirm with a status of ROUTE_ERROR.
   b. No further processing SHALL be done.
5. If an entry is found, do the following.
   a. Using the value of the MacInterfaceIndex of the nwkNeighborTable entry lookup the corresponding index in the nwkMacInterfaceTable.
   b. If the Enabled field of the nwkMacInterfaceTable entry has a value of FALSE, do the following.
      i. The NLDE SHALL issue an NLDE-DATA.confirm primitive with a status value of INVALID_INTERFACE.
      ii. No further processing SHALL be done.
6. If the DstAddr is set to All Devices in PAN (0xFFFF), macRXOnWhenIdle=TRUE (0xFFFD) or All routers and coordinators (0xFFFC) then examine that entry in the nwkMacInterfaceTable.
   a. If RoutersAllowed is set to TRUE, do the following.
      i. Issue an MCPS-DATA.request with the following parameters.
         1. SrcAddrMode SHALL be set to SHORT.

2. DstADdrMode SHALL be set to SHORT.
3. DstAddr SHALL be set to 0xFFFF.
4. IndirectTX SHALL be set to FALSE.

b. If RoutersAllowed is set to FALSE, keep processing.

7. Examine the nwkNeighborTable. For each device where macRxOnWhenIdle=FALSE, perform the following:
   a. Issue an MCPS-DATA.request with the following values.
      i. SrcAddrMode SHALL be set to SHORT.
      ii. DstADdrMode SHALL be set to SHORT.
      iii. DstAddr SHALL be set to the value of the NetworkAddress in entry of the nwkNeighborTable.
      iv. IndirectTX SHALL be set to TRUE.

The NWK layer will retry unicast transmissions to avoid transient failures at the MAC layer. These retries will be delayed in time to avoid short term interference or collisions that cause all MAC retries to fail. Broadcast transmissions at the Network layer use a passive acknowledgement mechanism to verify transmission.

On receipt of the MCPS-DATA.confirm primitive on a unicast the NLDE SHALL examine the status. If the MCPS-DATA status indicates a NO_ACK, TRANSACTION_OVERFLOW or a CHANNEL_ACCESS_FAILURE, the NLDE SHALL issue additional MCPS-DATA.request attempts up to *nwkcUnicastRetries*. Each attempt SHALL be delayed by at least nwkcUnicastRetryDelay and SHALL be re-encrypted with the newest network frame counter. After a MCPS-DATA.confirm is generated indicating success or all retries are exhausted the NLDE issues the NLDE-DATA.confirm primitive with a status equal to the last received status from the MAC sub-layer.

On receipt of a MCPS-DATA.confirm primitive from a broadcast, the NLDE immediately issues the NLDE-DATA.confirm primitive with the resulting status.

If the *nwkSecurityLevel* NIB attribute has a non-zero value and the SecurityEnable parameter has a value of TRUE, then NWK layer security processing will be applied to the frame before transmission as described in section 4.3. Otherwise, no security processing will be performed at the NWK layer for this frame. The security processing SHALL always be performed using device's own extended 64-bit IEEE address and Outgoing Frame Counter attribute of the NIB, and those values SHALL be put into the auxiliary NWK header of the frame, even if UseAlias parameter has a value of TRUE. If security processing is performed and it fails for any reason, then the frame is discarded and the NLDE issues the NLDE-DATA.confirm primitive with a Status parameter value equal to that returned by the security suite.

3.2.1.2 NLDE-DATA.confirm

This primitive reports the results of a request to transfer a data PDU (NSDU) from a local APS sub-layer entity to a single peer APS sub-layer entity.

3.2.1.2.1 Semantics of the Service Primitive

The semantics of this primitive are as follows:

```
NLDE-DATA.confirm        {
                         Status
                         NsduHandle,
                         TxTime
                         }
```

Table 3-3 specifies the parameters for the NLDE-DATA.confirm primitive.

Table 3-3. NLDE-DATA.confirm Parameters

Name	Type	Valid Range	Description
Status	Status	INV_REQUESTTYPE, MAX_FRM_COUNTER, NO_KEY, BAD_CCM_OUTPUT, ROUTE_ERROR, BT_TABLE_FULL, FRAME_NOT_BUFFERED or any status values returned from security suite or the MCPS-DATA.confirm primitive (see [B1]).	The status of the corresponding request.
NsduHandle	Integer	0x00 – 0xff	The handle associated with the NSDU being confirmed.
TxTime	Integer	Implementation specific	A time indication for the transmitted packet based on the local clock. The time SHOULD be based on the same point for each transmitted packet in a given implementation. This value is only provided if *nwkTimeStamp* is set to TRUE.

3.2.1.2.2 When Generated

This primitive is generated by the local NLDE in response to the reception of an NLDE-DATA.request primitive.

The Status field will reflect the status of the corresponding request, as described in section 3.2.1.1.3.

3.2.1.2.3 Effect on Receipt

On receipt of this primitive, the APS sub-layer of the initiating device is notified of the result of its request to transmit. If the transmission attempt was successful, the Status parameter will be set to SUCCESS. Otherwise, the Status parameter will indicate the error.

3.2.1.3 NLDE-DATA.indication

This primitive indicates the transfer of a data PDU (NSDU) from the NWK layer to the local APS sub-layer entity.

3.2.1.3.1 Semantics of the Service Primitive

The semantics of this primitive are as follows:

```
NLDE-DATA.indication        {
                            DstAddrMode,
                            DstAddr,
                            SrcAddr,
                            NsduLength,
                            Nsdu,
                            LQA*,
                            RxTime,
                            SecurityUse
                            }
```

Table 3-4 specifies the parameters for the NLDE-DATA.indication primitive.

Table 3-4. NLDE-DATA.indication Parameters

Name	**Type**	**Valid Range**	**Description**
DstAddrMode	Integer	0x02	The type of destination address supplied by the DstAddr parameter. This MAY have the following value: 0x02=16-bit network address of a device or a 16-bit broadcast address
DstAddr	16-bit Address	0x0000 –0xffff	The destination address to which the NSDU was sent.
SrcAddr	16-bit Device address	Any valid device address except a broadcast address	The individual device address from which the NSDU originated.
NsduLength	Integer	0 to *aMaxPHYPacketSize* – (*nwkcMACFrameOverhead* + *nwkcMinHeaderOverhead*)	The number of octets comprising the NSDU being indicated.
Nsdu	Set of octets	–	The set of octets comprising the NSDU being indicated.
LQA (LQI)	Integer	0x00 – 0xff	The estimated link quality for RF transmission from this device. See section 3.6.3 for a discussion of how this is calculated. This field SHALL be present in every neighbor table entry. When Active Power Control is used on this link, LQI SHALL be used instead of LQA.

Name	Type	Valid Range	Description
RxTime	Integer	Implementation specific	A time indication for the received packet based on the local clock. The time SHOULD be based on the same point for each received packet on a given implementation. This value is only provided if *nwkTimeStamp* is set to TRUE.
SecurityUse	Boolean	TRUE or FALSE	An indication of whether the received data frame is using security. This value is set to TRUE if security was applied to the received frame or FALSE if the received frame was unsecured.

3.2.1.3.2 When Generated

This primitive is generated by the NLDE and issued to the APS sub-layer on receipt of an appropriately addressed data frame from the local MAC sub-layer entity.

3.2.1.3.3 Effect on Receipt

On receipt of this primitive, the APS sub-layer is notified of the arrival of data at the device.

3.2.2 NWK Management Service

The NWK layer management entity SAP (NLME-SAP) allows the transport of management commands between the next higher layer and the NLME. Table 3-5 lists the primitives supported by the NLME through the NLME-SAP interface and the sections containing details on each of these primitives.

Table 3-5. Summary of the Primitives Accessed Through the NLME-SAP

	Section Number in this Specification			
Name	Request	Indication	Response	Confirm
NLME-NETWORK-AND-PARENT-DISCOVERY	3.2.2.3			3.2.2.4
NLME-NETWORK-FORMATION	3.2.2.5			3.2.2.6
NLME-PERMIT-JOINING	3.2.2.7			3.2.2.8
NLME-START-ROUTER	3.2.2.9			3.2.2.10
NLME-ED-SCAN	3.2.2.11			3.2.2.12
NLME-JOIN	3.2.2.13	3.2.2.14		3.2.2.15
NLME-ADD-NEIGHBOR	3.2.2.16			3.2.2.17

Name	Section Number in this Specification			
	Request	Indication	Response	Confirm
NLME-LEAVE	3.2.2.18	3.2.2.19		3.2.2.20
NLME-RESET	3.2.2.21			3.2.2.22
NLME-SYNC	3.2.2.24			3.2.2.25
NLME-GET	3.2.2.27			3.2.2.28
NLME-SET	3.2.2.29			3.2.2.30
NLME-NWK-STATUS		3.2.2.31		
NLME-ROUTE-DISCOVERY	3.2.2.32			3.2.2.33
NLME-SET-INTERFACE	3.2.2.34			3.2.2.35
NLME-GET-INTERFACE	3.2.2.36			3.2.2.37

3.2.2.1 MAC Interfaces

The NWK layer MAY optionally support one or more MAC interface. Where there are multiple MAC interfaces to a single NWK layer, they SHALL all use the same PANID. Multi-MAC Devices SHALL use the same EUI64 and short address on all MAC interfaces. These interfaces MAY be enabled or disabled independently. At least one entry in the nwkMacInterfaceTable SHALL have an Enabled field set to TRUE for the network layer to be considered formed or joined.

3.2.2.2 Network Management Data Structures

The following network management data structures are utilized by the NLME primitives.

3.2.2.2.1 Channel List Structure

To convey channel information for a device that MAY be operating on multiple MAC interfaces it is necessary to utilize a data structure that can convey this complete information. Channels are divided into groups known as pages. Each page indicates information about a particular band while the list of channels is indicated via bits within the page. A ChannelList structure SHALL contain only one instance of each channel page. Table 3-6 describes the ChannelList structure.

Table 3-6. Field Descriptions of the ChannelList Structure

Name	Type	Valid Range	Description
Channel Page Count	Integer	0 – 255	The number of Channel Page Structures contained within the Channel List Structure
Channel Page Structure	Channel Page Structure	Variable	The set of channels for this channel page. See Table 3-7.

Table 3-7 describes the fields of the Channel Page structure.

Table 3-7. Field Descriptions of the Channel Page Structure

Name	Type	Valid Range	Description
Channels Field	Bitmap	32-bit field	The five most significant bits (b27,..., b31) represent the binary encoded Channel Page. The 27 least significant bits (b0, b1,... b26) indicate which channels are to be scanned (1 = scan, 0 = do not scan) for each of the 27 valid channels

Annex D defines the list of valid channel pages and their frequency bands.

Table 3-8. Over-the-air Format of the Channel List Structure

Octets: 1	Octets: 4	Octets: 0/4	Octets: 0/4…
Channel Page Count	First Channel Page	n Channel Page	n+1 Channel Page

Note: 32-bit Channel Page structures are encoded as little-endian values in over-the-air frames.

3.2.2.2.2 Validating the List of Channels

Any primitive that accepts a Channel List Structure SHALL validate the list of channels is acceptable. For each channel specified in the primitive's Channel List Structure do the following. For each entry in the *nwkMacInterfaceTable* where the Enabled field of the interface is set to TRUE, examine if the Channel page and Channel passed to the primitive corresponds to a Channel Page and Channel enumerated in the Channel List Structure of the *nwkMacInterfaceTable* entry. If all entries have been examined and the channel does not match then processing SHALL fail for that primitive.

3.2.2.2.3 Energy Detect List Structure

The energy detect list structure is used to convey energy values for each channel that was scanned. Table 3-9 indicates the format.

Table 3-9. Field Descriptions of the EnergyDetectListStructure

Name	Type	Valid Range	Description
Channel Count	Integer	0 – 255	The number of EnergyDetectChannelInfo items in the EnergyDetectListStructure, which is the total number of channels scanned.
ChannelInfo	EnergyDetectChannelInfo	Variable	See Table 3-10.

Table 3-10 describes the format of the EnergyDetectChannelInfo.

Table 3-10. Field Descriptions of the EnergyDetectChannelInfo

Name	Type	Valid Range	Description
ChannelPage	Integer	0 - 31	The channel page of the channel that was scanned.
ChannelNumber	Integer	0 – 26	The channel number within the channel page.
EnergyDetected	Integer	0 – 255	The energy measurement.

3.2.2.3 NLME-NETWORK-AND-PARENT-DISCOVERY.request

This primitive allows the next higher layer to request that the NWK layer discover networks currently operating within the POS.

3.2.2.3.1 Semantics of the Service Primitive

The semantics of this primitive are as follows:

```
NLME-NETWORK-AND-PARENT-DISCOVERY.request    {
                                              ScanChannelsListStructure,
                                              ScanDuration,
                                              OnlyPermitJoinNetworks,
                                              OnlyEndDeviceCapacity
                                              }
```

Table 3-11 specifies the parameters for the NLME-NETWORK-AND-PARENT-DISCOVERY.request primitive.

Table 3-11. NLME-NETWORK-And-PARENT-DISCOVERY.request Parameters

Name	Type	Valid Range	Description
ScanChannelsListStructure	Channel List Structure	Varies	A list of channel pages and the channels within those pages that the discovery SHALL be performed upon.
ScanDuration	Integer	0x00 – 0x0e	A value used to calculate the length of time to spend scanning each channel: The time spent scanning each channel is (*aBaseSuperframeDuration* * $(2^n + 1)$) symbols, where n is the value of the ScanDuration parameter. For more information on MAC sub-layer scanning (see [B1]).
OnlyPermitJoinNetworks	Boolean	TRUE or FALSE	This indicates that only beacons with Association Permit set to TRUE SHALL be considered as parents.
OnlyEndDeviceCapacity	Boolean	TRUE or FALSE	This indicates that only beacons with TRUE for the End Device Capacity field of the Zigbee Beacon Info field SHALL be considered.

3.2.2.3.2 When Generated

This primitive is generated by the next higher layer of a Zigbee device and issued to its NLME to request the discovery of networks operating within the device's personal operating space (POS).

3.2.2.3.3 Effect on Receipt

For each interface in the nwkMacInterfaceTable perform the following:

1. Determine if the SupportedChannels value in the entry contains channels that are to be scanned, as specified in the ScanChannelsListStructure . If no channels are supported, return INVALID_PARAMETER and no further processing SHALL be done.
2. If a channel in the interface is to be scanned, perform an MLME-SCAN.request as follows:
   i. Set the ScanType of MLME ScanType parameter to the ScanType of the nwkMacInterfaceTable entry.
   ii. Set the ScanChannels to the intersection of the channels of the SupportedChannels item in the nwkMacInterfaceTable entry and the ScanChannelsListStructure passed to this primitive.
   iii. For each MLME-SCAN.confirm primitive the beacons SHALL be processed according to section 3.6.1.3 Network and Parent Discovery.

On receipt of the last MLME-SCAN.confirm primitive, the NLME issues the NLME-NETWORK-AND-PARENT-DISCOVERY.confirm primitive containing the information about the discovered networks with a Status parameter value equal to that returned with the MLME-SCAN.confirm.

3.2.2.4 NLME-NETWORK-AND-PARENT-DISCOVERY.confirm

This primitive reports the results of a network and parent discovery operation. Details of the networks and parents discovered can be retrieved via the NLME-GET.request for the *nwkDiscoveryTable* NIB attribute.

3.2.2.4.1 Semantics of the Service Primitive

The semantics of this primitive are as follows:

```
NLME-NETWORK-AND-PARENT-DISCOVERY.confirm        {
                                                  Status
                                                  }
```

Table 3-12 describes the arguments of the NLME-NETWORK-AND-PARENT-DISCOVERY.confirm primitive.

Table 3-12. NLME-NETWORK-AND-PARENT-DISCOVERY.confirm Parameters

Name	Type	Valid Range	Description
Status	Status	Any status value returned with the MLME-SCAN.confirm primitive.	See [B1].

3.2.2.4.2 When Generated

On receipt of all the MLME-SCAN.confirm primitive(s), the NLME issues the NLME-NETWORK-AND-PARENT-DISCOVERY.confirm primitive containing the Status parameter value equal to that returned with the MLME-SCAN.confirm.

3.2.2.4.3 Effect on Receipt

On receipt of this primitive, the next higher layer is notified of the results of a network search. The next higher layer MAY use the NLME-GET.request primitive to retrieve the results stored in the nwkDiscoveryTable of the NIB.

3.2.2.5 NLME-NETWORK-FORMATION.request

This primitive allows the next higher layer to request that the device start a new Zigbee network with itself as the coordinator and subsequently make changes to its superframe configuration.

3.2.2.5.1 Semantics of the Service Primitive

The semantics of this primitive are as follows:

```
NLME-NETWORK-FORMATION.request          {
                                        ScanChannelsListStructure,
                                        ScanDuration,
                                        BeaconOrder,
                                        SuperframeOrder,
                                        BatteryLifeExtension
                                        DistributedNetwork
                                        DistributedNetworkAddress

                                        }
```

Table 3-13 specifies the parameters for the NLME-NETWORK-FORMATION.request primitive.

Table 3-13 NLME-NETWORK-FORMATION.request Parameters

Name	Type	Valid Range	Description
ScanChannelsListStructure	Channel List Structure	Varies	The list of all channel pages and the associated channels that SHALL be scanned.
ScanDuration	Integer	0x00 – 0x0e	A value used to calculate the length of time to spend scanning each channel. The time spent scanning each channel is (*aBaseSuperframeDuration* * (2^n + 1)) symbols, where *n* is the value of the ScanDuration parameter (see [B1]).
BeaconOrder	Integer	0x00 – 0x0f	The beacon order of the network that the higher layers wish to form.
SuperframeOrder	Integer	0x00 – 0x0f	The superframe order of the network that the higher layers wish to form.
BatteryLifeExtension	Boolean	TRUE or FALSE	If this value is TRUE, the NLME will request that the Zigbee coordinator is started supporting battery life extension mode; If this value is FALSE, the NLME will request that the Zigbee coordinator is started without supporting battery life extension mode.
DistributedNetwork	Boolean	TRUE or FALSE	If this value is TRUE then it indicates that distributed network security will be used and therefore it is permissible for a Zigbee router to form the network. If FALSE, then this primitive MAY only be called by the Zigbee Coordinator.

Name	Type	Valid Range	Description
DistributedNetwork-Address	Integer	0x0001 – 0xFFF7	The address the device will use when forming a distributed network.

3.2.2.5.2 When Generated

This primitive is generated by the next higher layer of a Zigbee coordinator-capable device and issued to its NLME to request the initialization of itself as the Zigbee coordinator of a new network.

3.2.2.5.3 Effect on Receipt

1. If DistributedNetwork is set to FALSE and the device is not capable of being a Zigbee coordinator, the NLME SHALL issue the NLME-NETWORK-FORMATION.confirm primitive with the Status parameter set to INV_REQUESTTYPE.
2. If DistributedNetwork is set to TRUE and the device is not capable of being a Zigbee router then NLME issues the NLME-NETWORK-FORMATION.confirm primitive with the Status parameter set to INV_REQUESTTYPE.
3. If DistributedNetwork is set to TRUE and the DistributedNetworkAddress is outside the valid range then processing SHALL fail with the Status parameter set to INV_REQUESTTYPE.
4. On receipt of this primitive the NLME SHALL first validate the ChannelListStructure parameter according to section 3.2.2.2.2. If validation fails the NLME-NETWORK-FORMATION.confirm primitive SHALL be issued with a Status parameter set to INVALID_PARAMETER.
5. Determine the corresponding interface entries in the nwkMacInterfaceTable where the SupportedChannels parameter indicates support for the channels that were indicated within the ScanChannelsListStructure. If multiple Channel Pages are indicated the scan SHALL be repeated for each page. When multiple channels are provided to the NLME-FORMATION.request it is recommended to perform an energy detection scan to determine the channel with the lowest energy. After selecting that channel, it is required to pick a random PANID and recommended to perform an active scan to ensure it does not conflict with any of the currently operating 802.15.4 PANs.
6. Using those interfaces' MLME, do the following.
    a. Initiate one or more MLME-SCAN.request primitives with the following parameters.
        i. ScanType SHALL be set to ED.
        ii. ScanChannels SHALL be set to the channels indicated in the NLME ScanChannelsListStructure parameter for the relevant page.
        iii. ChannelPage SHALL be set to the corresponding channel page indicated in the NLME ScanChannelsListStructure parameter.
        iv. ScanDuration SHALL be set to the value indicated in the NLME ScanDuration parameter.
    b. Upon receipt of each MLME-SCAN.confirm, do the following:
        i. If the MLME-SCAN.confirm status does not indicate SUCCESS, issue an NLME-NETWORK-FORMATION.confirm with the corresponding Status returned by the MLME. No further processing of the NLME SHALL be done.
        ii. If the Status indicates SUCCESS, pick a list of acceptable channels on which to perform an active scan.
    c. Initiate one or more MLME-SCAN.request primitives with the following parameters.
        i. ScanType SHALL be set to ACTIVE.
        ii. ScanChannels SHALL be set to the acceptable channels indicated in step b ii above
        iii. ChannelPage SHALL be set to the corresponding channel page indicated in step b ii above

iv. ScanDuration SHALL be set to the value indicated in the NLME ScanDuration parameter.

d. Upon receipt of each MLME-SCAN.confirm, do the following.

i. If the Status does not indicate SUCCESS, or NO_BEACON, issue an NLME-NETWORK-FORMATION.confirm with the corresponding Status returned by the MLME. No further processing of the NLME SHALL be done.

ii. After receipt of the last MLME-SCAN.confirm, if the Status indicates SUCCESS or NO_BEACON, review the list of returned PAN descriptors and, for each interface identified in 5. above, find the first channel with the lowest number of existing networks on which to form a network, favoring a channel with no detected networks. Pick a random PAN Identifier that does not match any of the values returned in the PANDescriptorList of the MLME-SCAN.confirm for the selected channels.

e. If no suitable channel or PAN identifier can be found, the NLME issues the NLME-NETWORK-FORMATION.confirm primitive with the Status parameter set to STARTUP_FAILURE.

f. Once suitable channel(s) and PAN identifier are found, an address SHALL be chosen and the MAC sub-layer informed of the resultant address.

i. Issue an MLME-SET.request for the MIB value macShortAddress.

1. If the DistributedNetwork parameter is set to FALSE, set the MIB value to 0x0000.

2. If the DistributedNetwork parameter is set to TRUE, set the MIB value to the DistributedNetworkAddress.

g. If the NIB attribute nwkExtendedPANId is equal to 0x0000000000000000, do the following:

i. Perform an MLME-GET.request for the value macExtendedAddress.

ii. Set the nwkExtendedPANId equal to the value of macExtendedAddress.

h. The device SHALL issue an MLME-START.request to the appropriate MAC sub-layers.

i. If DistributedNetwork is FALSE, set PANCoordinator parameter to TRUE. Otherwise, set PANCoordinator to FALSE.

ii. Set BeaconOrder, SuperframeOrder, and BatteryLifeExtension parameters to the values indicated in the NLME-NETWORK-FORMATION.request.

iii. CoordRealignment parameter SHALL be set to FALSE.

i. On receipt of the last MLME-START.confirm primitive, issue an NLME-NETWORK-FORMATION.confirm primitive to the next higher layer setting the Status to the Status value returned by the MLME-START.confirm primitive.

Note: It is possible for the application to delay enabling additional MAC interfaces until it is necessary to do so. This MAY be done to conserve bandwidth or for other administrative reasons.

3.2.2.6 NLME-NETWORK-FORMATION.confirm

This primitive reports the results of the request to initialize a Zigbee coordinator in a network.

3.2.2.6.1 Semantics of the Service Primitive

The semantics of this primitive are as follows:

```
NLME-NETWORK-FORMATION.confirm      {
                                    Status
                                    }
```

Table 3-14 specifies the parameters for the NLME-NETWORK-FORMATION.confirm primitive.

Table 3-14. NLME-NETWORK-FORMATION.confirm Parameters

Name	Type	Valid Range	Description
Status	Status	INV_REQUESTTYPE, STARTUP_FAILURE, or any status value returned from the MLME-START.confirm primitive.	The result of the attempt to initialize a Zigbee coordinator.

3.2.2.6.2 When Generated

This primitive is generated by the NLME and issued to its next higher layer in response to an NLME-NETWORK-FORMATION.request primitive. This primitive returns a status value of INV_REQUESTTYPE, STARTUP_FAILURE or any status value returned from the MLME-START.confirm primitive. Conditions under which these values MAY be returned are described in section 3.2.2.5.3.

3.2.2.6.3 Effect on Receipt

On receipt of this primitive, the next higher layer is notified of the results of its request to initialize the device as a Zigbee coordinator. If the NLME has been successful, the Status parameter will be set to SUCCESS. Otherwise, the Status parameter indicates the error.

3.2.2.7 NLME-PERMIT-JOINING.request

This primitive allows the next higher layer of a Zigbee coordinator or router to set its MAC sub-layer association permit flag for a fixed period when it MAY accept devices onto its network.

3.2.2.7.1 Semantics of the Service Primitive

The semantics of this primitive are as follows:

```
NLME-PERMIT-JOINING.request      {
                                 PermitDuration
                                 }
```

Table 3-15 specifies the parameters for the NLME-PERMIT-JOINING.request primitive.

Table 3-15. NLME-PERMIT-JOINING.request Parameters

Name	Type	Valid Range	Description
PermitDuration	Integer	0x00 – 0xff	The length of time in seconds during which the Zigbee coordinator or router will allow associations. The value 0x00 and 0xff indicate that permission is disabled or enabled, respectively, without a specified time limit.

3.2.2.7.2 When Generated

This primitive is generated by the next higher layer of a Zigbee coordinator or router and issued to its NLME whenever it would like to permit or prohibit the joining of the network by new devices.

3.2.2.7.3 Effect on Receipt

It is only permissible that the next higher layer of a Zigbee coordinator or router issue this primitive. On receipt of this primitive by the NWK layer of a Zigbee end device, the NLME-PERMIT-JOINING.confirm primitive returns a status of INV_REQUESTTYPE.

On receipt of this primitive with the PermitDuration parameter set to 0x00, the NLME sets the MAC PIB attribute, *macAssociationPermit*, to FALSE by issuing the MLME-SET.request primitive to the MAC sub-layer. Once the MLME-SET.confirm primitive is received, the NLME issues the NLME-PERMIT-JOINING.confirm primitive with a status equal to that received from the MAC sub-layer.

On receipt of this primitive with the PermitDuration parameter set to 0xff, the NLME sets the MAC PIB attribute, *macAssociationPermit*, to TRUE by issuing the MLME-SET.request primitive to the MAC sub-layer. Once the MLME-SET.confirm primitive is received, the NLME issues the NLME-PERMIT-JOINING.confirm primitive with a status equal to that received from the MAC sub-layer.

On receipt of this primitive with the PermitDuration parameter set to any value other than 0x00 or 0xff, the NLME sets the MAC PIB attribute, *macAssociationPermit*, to TRUE as described above. Following the receipt of the MLME-SET.confirm primitive, the NLME starts a timer to expire after PermitDuration seconds. Once the timer is set, the NLME issues the NLME-PERMIT-JOINING.confirm primitive with a status equal to that received by the MAC sub-layer. On expiration of the timer, the NLME sets *macAssociationPermit* to FALSE by issuing the MLME-SET.request primitive.

Every NLME-PERMIT-JOINING.request primitive issued by the next higher layer supersedes all previous requests.

3.2.2.8 NLME-PERMIT-JOINING.confirm

This primitive allows the next higher layer of a Zigbee coordinator or router to be notified of the results of its request to permit the acceptance of devices onto the network.

3.2.2.8.1 Semantics of the Service Primitive

The semantics of this primitive are as follows:

```
NLME-PERMIT-JOINING.confirm     {
                                Status
                                }
```

Table 3-16 specifies the parameters for the NLME-PERMIT-JOINING.confirm primitive.

Table 3-16. NLME-PERMIT-JOINING.confirm Parameters

Name	Type	Valid Range	Description
Status	Status	INV_REQUESTTYPE or any status returned from the MLME-SET.confirm primitive (see [B1]).	The status of the corresponding request

3.2.2.8.2 When Generated

This primitive is generated by the initiating NLME of a Zigbee coordinator or router and issued to its next higher layer in response to an NLME-PERMIT-JOINING.request. The Status parameter either indicates the status received from the MAC sub-layer or an error code of INV_REQUESTTYPE. The reasons for these status values are described in section 3.2.2.7.

3.2.2.8.3 Effect on Receipt

On receipt of this primitive, the next higher layer of the initiating device is notified of the results of its request to permit devices to join the network.

3.2.2.9 NLME-START-ROUTER.request

This primitive allows the next higher layer of a Zigbee router to initiate the activities expected of a Zigbee router including the routing of data frames, route discovery, and the accepting of requests to join the network from other devices.

3.2.2.9.1 Semantics of the Service Primitive

The semantics of this primitive are as follows:

```
NLME-START-ROUTER.request          {
                                   BeaconOrder,
                                   SuperframeOrder,
                                   BatteryLifeExtension
                                   }
```

Table 3-17 specifies the parameters for NLME-START-ROUTER.request.

Table 3-17. NLME-START-ROUTER.request Parameters

Name	Type	Valid Range	Description
BeaconOrder	Integer	0x00 – 0x0f	The beacon order of the network that the higher layers wish to form.
SuperframeOrder	Integer	0x00 – 0x0f	The superframe order of the network that the higher layers wish to form.
BatteryLifeExtension	Boolean	TRUE or FALSE	If this value is TRUE, the NLME will request that the Zigbee router is started supporting battery life extension mode. If this value is FALSE, the NLME will request that the Zigbee router is started without supporting battery life extension mode.

3.2.2.9.2 When Generated

This primitive is generated by the next higher layer of a new device and issued to its NLME to request the initialization of itself as a Zigbee router.

3.2.2.9.3 Effect on Receipt

On receipt of this primitive by a device that is not already joined to a Zigbee network as a router, the NLME issues the NLME-START-ROUTER.confirm primitive with the Status parameter set to INV_REQUESTTYPE.

On receipt of this primitive by the NLME of a device that is joined to a Zigbee network as a router, the NLME SHALL issue the MLME-START.request primitive to each MAC sub-layer entry in the nwkMacInterfaceTable where the Enabled element is set to TRUE. The BeaconOrder, SuperframeOrder, and BatteryLifeExtension parameters of the MLME-START.request primitive will have the values given by the corresponding parameters of the NLME-START-ROUTER.request. The CoordRealignment parameter in the MLME-START.request primitive is set to FALSE if the primitive is issued to start as a router for the first time. The CoordRealignment parameter is set to TRUE thereafter if the primitive is issued to change any of the PAN configuration attributes.

On receipt of the associated MLME-START.confirm primitive, the NLME issues the NLME-START-ROUTER.confirm primitive to the next higher layer with the status returned from the MLME-START.confirm primitive. If, and only if, the status returned from the MLME-START.confirm primitive is SUCCESS, the device MAY then begin to engage in the activities EXPECTED of a Zigbee router including the routing of data frames, route discovery, and the accepting of requests to join the network from other devices. Otherwise, the device is expressly forbidden to engage in these activities.

Note after a router has gone through a joining process there MAY be interfaces that have not associated. Those remaining interfaces still need to select a suitable channel for operation. Those will need to execute the normal channel selection process (see network formation text) before issuing the MLME-START.request. See section 3.6.12.2 for starting a Multi-Mac interface device.

3.2.2.10 NLME-START-ROUTER.confirm

This primitive reports the results of the request to initialize a Zigbee router.

3.2.2.10.1 Semantics of the Service Primitive

The semantics of this primitive are as follows:

```
NLME-START-ROUTER.confirm        {
                                 Status
                                 }
```

Table 3-18 specifies the parameters for NLME-START-ROUTER.confirm.

Table 3-18. NLME-START-ROUTER.confirm Parameters

Name	Type	Valid Range	Description
Status	Status	INV_REQUESTTYPE or any status value returned from the MLME-START.confirm primitive.	The result of the attempt to initialize a Zigbee router.

3.2.2.10.2 When Generated

This primitive is generated by the NLME and issued to its next higher layer in response to an NLME-START-ROUTER.request primitive. This primitive returns a status value of INV_REQUESTTYPE or any status value returned from the MLME-START.confirm primitive. Conditions under which these values MAY be returned are described in section 3.2.2.9.3.

3.2.2.10.3 Effect on Receipt

On receipt of this primitive, the next higher layer is notified of the results of its request to initialize a Zigbee router. If the NLME has been successful, the Status parameter will be set to SUCCESS. Otherwise, the Status parameter indicates the error.

3.2.2.11 NLME-ED-SCAN.request

This primitive allows the next higher layer to request an energy scan to evaluate channels in the local area.

3.2.2.11.1 Semantics of the Service Primitive

The semantics of this primitive are as follows:

NLME-ED-SCAN.request	{
	ScanChannelsListStructure,
	ScanDuration,
	}

Table 3-19 specifies the parameters for the service primitive.

Table 3-19. NLME-ED-SCAN.request Parameters

Name	Type	Valid Range	Description
ScanChannelsListStructure	Channel-ListStructure	Varies	The list of all channel pages and the associated channels that SHALL be scanned.
ScanDuration	Integer	0x00 – 0x0e	A value used to calculate the length of time to spend scanning each channel. The time spent scanning each channel is (*aBaseSuperframe-Duration* * (2^n + 1)) symbols, where *n* is the value of the ScanDuration parameter (see [B1]).

3.2.2.11.2 When Generated

The next higher layer of a device generates this primitive to request to conduct an energy scan of channels.

3.2.2.11.3 Effect on Receipt

On receipt the NLME SHALL first validate the ScanChannelsListStructure parameter according to section 3.2.2.2.2. If validation fails the NLME-ED-SCAN.confirm primitive SHALL be issued with a Status parameter set to INVALID_PARAMETER.

If the device is currently joined to a network, the device will temporarily stop operating on the network to conduct an energy scan.

The NLME SHALL issue the MLME-SCAN.request primitive to every valid channel page in each MAC sub-layer that has a nwkMacInterfaceTable entry where the SupportedChannels field corresponds to a bit in the ScanChannelsListStructure. The MLME-SCAN.request SHALL set the ScanType parameter to indicate an energy detection scan (ED) and the ScanChannelsListStructure and ScanDuration SHALL be set to the values passed from the NLME request.

3.2.2.12 NLME-ED-SCAN.confirm

This primitive provides the next higher layer results from an energy scan.

3.2.2.12.1 Semantics of the Service Primitive

The semantics of this primitive are as follows:

NLME-ED-SCAN.confirm	{
	Status
	EnergyDetectListStructure
	}

Table 3-20 specifies the parameters for the service primitive.

Table 3-20. NLME-ED-SCAN.confirm

Name	Type	Valid Range	Description
Status	Status	SUCCESS, or any valid code from the MAC	The status of the request.
EnergyDetect-ListStructure	Ener-gyDetect-ListStructure	Varies	The list of energy measurements in accordance with Table 3-9.

3.2.2.12.2 When Generated

This primitive is generated by the NLME of a Zigbee device in response to an NLME-ED-SCAN.request. The Status indicates the Status received from the MAC sub-layer primitive MLME-SCAN.confirm. The NLME SHALL construct an EnergyDetectListStructure that contains all received EnergyDetectList values of the MLME-SCAN.confirm primitives that were generated by the corresponding NLME-ED-SCAN.request.

3.2.2.12.3 Effect on Receipt

On receipt of this primitive, the next higher layer is notified of the results of an ED scan.

3.2.2.13 NLME-JOIN.request

This primitive allows the next higher layer to request to join or rejoin a network, or to change the operating channel for the device while within an operating network.

3.2.2.13.1 Semantics of the Service Primitive

The semantics of this primitive are as follows:

NLME-JOIN.request	{
	ExtendedPANId,
	RejoinNetwork,
	ScanChannelsListStructure,
	ScanDuration,
	CapabilityInformation,
	SecurityEnable
	}

Table 3-21 specifies the parameters for the NLME-JOIN.request primitive.

Table 3-21. NLME-JOIN.request

Name	Type	Valid Range	Description
ExtendedPANId	Integer	0x0000000000000001 – 0xfffffffffffffffe	The 64-bit PAN identifier of the network to join.
RejoinNetwork	Integer	0x00 – 0x03	This parameter controls the method of joining the network. The parameter is 0x00 if the device is requesting to join a network through association. The parameter is 0x01 if the device is joining directly or rejoining the network using the orphaning procedure. The parameter is 0x02 if the device is joining the network using the NWK rejoining procedure. The parameter is 0x03 if the device is to change the operational network channel to that identified in the ScanChannels parameter.
ScanChannelsListStructure	Channel List Structure	Varies	The list of all channel pages and the associated channels that SHALL be scanned.
ScanDuration	Integer	0x00-0x0e	A value used to calculate the length of time to spend scanning each channel. The time spent scanning each channel is (*aBaseSuperframeDuration* * $(2^n + 1)$) symbols, where *n* is the value of the ScanDuration parameter (see [B1]).
CapabilityInformation	Bitmap	See Table 3-67.	The operating capabilities of the device being directly joined.
SecurityEnable	Boolean	TRUE or FALSE	If the value of RejoinNetwork is 0x02 and this is TRUE than the device will try to rejoin securely. Otherwise, this is set to FALSE.

3.2.2.13.2 When Generated

The next higher layer of a device generates this primitive to request to:

- Join a network using the MAC association procedure.
- Join or rejoin a network using the orphaning procedure.
- Join or rejoin a network using the NWK rejoin procedure.
- Switch the operating channel for a device that is joined to a network.

3.2.2.13.3 Effect on Receipt

On receipt the NLME SHALL first validate the ChannelListStructure according to section 3.2.2.2.2. If validation fails the NLME-JOIN.confirm primitive SHALL be issued with a Status parameter set to INVALID_PARAMETER.

When performing a join (Rejoin parameter is set to 0x00), the device SHALL verify that the ChannelListStructure indicates a single channel selected for the join. If more than a single channel has been selected, the NLME-JOIN.request primitive SHALL be issued with a Status parameter set to INVALID_PARAMETER. When performing a rejoin (Rejoin parameter not equal to 0x00) the ChannelListStructure MAY indicate multiple channels to try.

On receipt of this primitive by a device that is currently joined to a network and with the RejoinNetwork parameter equal to 0x00, the NLME issues an NLME-JOIN.confirm primitive with the Status parameter set to INV_REQUESTTYPE.

On receipt of this primitive by a device that is not currently joined to a network and with the RejoinNetwork parameter equal to 0x00, the device attempts to join the network specified by the 64-bit ExtendedPANId parameter as described in section 3.6.1.6.1.1.

Whether joining or rejoining, the device SHALL set the nwkParentInformation in the NIB to 0.

If a device receives this primitive and the RejoinNetwork parameter is equal to 0x01, then it attempts to join or rejoin the network using orphaning as described in section 3.6.1.6.1.2.

On receipt of this primitive with the RejoinNetwork parameter is equal to 0x02, the device attempts to rejoin the network with 64-bit extended PAN ID given by the ExtendedPANId parameter. The procedure for rejoining is given in section 3.6.1.6.1.2.

Once the device has successfully joined a network, it will set the value of the *nwkExtendedPANId* NIB attribute to the extended PAN identifier of the network to which it joined.

If a device receives this primitive and the RejoinNetwork parameter is equal to 0x03, and the device supports setting the value of phyCurrentChannel then the device attempts to switch the operating channel to that provided in the ScanChannels parameter. If more than one channel is provided in the ScanChannels parameter, the NLME issues an NLME-JOIN.confirm primitive with the Status parameter set to INV_REQUESTTYPE. If the channel change is performed successfully, then the device issues a NLME-JOIN.confirm with the Status parameter set to SUCCESS. If the device does not support the setting of phyCurrentChannel directly, then the NLME issues a NLME-JOIN.confirm primitive with the Status parameter set to UNSUPPORTED_ATTRIBUTE.

If the MAC layer returned an error status during the channel change then this status SHALL be reported in the status field of the NLME-JOIN.confirm primitive.

3.2.2.14 NLME-JOIN.indication

This primitive allows the next higher layer of a Zigbee coordinator or Zigbee router to be notified when a new device has successfully joined its network by association or rejoined using the NWK rejoin procedure as described in section 3.6.1.6.1.

3.2.2.14.1 Semantics of the Service Primitive

The semantics of this primitive are as follows:

```
NLME-JOIN.indication            {
                                InterfaceIndex,
                                NetworkAddress,
                                ExtendedAddress,
                                CapabilityInformation,
                                JoinerMethod,
                                JoiningDeviceTLVs
                            }
```

Table 3-22 specifies the parameters for the NLME-JOIN.indication primitive.

Table 3-22. NLME-JOIN.indication Parameters

Name	Type	Valid Range	Description
InterfaceIndex	Integer	0 – 31	The index of the interface in the Mac Interface Table to set.
NetworkAddress	Network address	0x0001 – 0xfff7	The network address of an entity that has been added to the network.
ExtendedAddress	64-bit IEEE address	Any 64-bit, IEEE address	The 64-bit IEEE address of an entity that has been added to the network.
CapabilityInformation	Bitmap	See [B1].	Specifies the operational capabilities of the joining device.
JoinerMethod	Enumeration	0x00 – 0x05	This indicates the mechanism used to join or rejoin. It has the following status values. 0 – MAC Association 1 – Network rejoin without security 2 – Secure network rejoin 3 – Network commissioning join without security 4 – Network commissioning rejoin without security 5 – Secure network commissioning rejoin
JoiningDeviceTLVs	TLVs	Varies	This is a set of TLVs communicated by the Joining Device to the parent router. via the Network Commissioning Request Command Frame. See section 3.4.14.3 for the set of TLVs that can be communicated.”

3.2.2.14.2 When Generated

This primitive is generated by the NLME of a Zigbee coordinator or router and issued to its next higher layer on successfully adding a new device to the network using the MAC association procedure as shown in Figure 3-45, or on allowing a device to rejoin the network using the NWK rejoining procedure as shown in Figure 3-46.

3.2.2.14.3 Effect on Receipt

On receipt of this primitive, the next higher layer of a Zigbee coordinator or Zigbee router is notified that a new device has joined its network.

3.2.2.15 NLME-JOIN.confirm

This primitive allows the next higher layer to be notified of the results of its request to join a network.

3.2.2.15.1 Semantics of the Service Primitive

The semantics of this primitive are as follows:

```
NLME-JOIN.confirm        {
                         Status,
                         NetworkAddress,
                         ExtendedPANID,
                         ChannelListStructure
                         Enhanced BeaconType
                         MacInterface Index,
                         JoinMethodUsed
                         }
```

Table 3-23 specifies the parameters for the NLME-JOIN.confirm primitive.

Table 3-23. NLME-JOIN.confirm

Name	Type	Valid Range	Description
Status	Status	INV_REQUESTTYPE, NOT_PERMITTED, NO_NETWORKS, or any status value returned from the MLME-ASSOCIATE.confirm primitive, the MLME-SCAN.confirm primitive or the PLME-SET.confirm	The status of the corresponding request.
NetworkAddress	Integer	0x0001 – 0xfff7 and 0xffff	The 16-bit network address that was allocated to this device. This parameter will be equal to 0xffff if the join attempt was unsuccessful.
ExtendedPANID	Integer	0x0000000000000001 – 0xfffffffffffffffe	The 64-bit extended PAN identifier for the network of which the device is now a member.
Channel List Structure	Channel-ListStructure	Varies	The structure indicating the current channel of the network that has been joined.
Enhanced BeaconType	Boolean	TRUE or FALSE	Returns TRUE if using Enhanced Beacons.
MacInterface Index	Integer	0 – 31	Value of Mac Interface Index from nwkMACInterfaceTable.

Name	Type	Valid Range	Description
JoinMethodUsed	Enumeration	0 – 31	0 – MAC Association 1 – Network rejoin without security 2 – Secure network rejoin 3 – Network commissioning join without security 4 – Network commissioning rejoin without security 5 – Secure network commissioning rejoin

3.2.2.15.2 When Generated

This primitive is generated by the initiating NLME and issued to its next higher layer in response to an NLME-JOIN.request primitive. If the request was successful, the Status parameter will have a value of SUCCESS. Otherwise, the Status parameter indicates an error code of INV_REQUESTTYPE, NOT_PERMITTED, NO_NETWORKS or any status value returned from either the MLME-ASSOCIATE.confirm primitive, the MLME-SCAN.confirm primitive or the PLME-SET.confirm primitive. The reasons for these status values are fully described in section 3.2.2.13.3.

3.2.2.15.3 Effect on Receipt

On receipt of this primitive, the next higher layer of the initiating device is notified of the results of its request to join a network using the MAC sub-layer association procedure, to join directly using the MAC sub-layer orphaning procedure, or to re-join a network once it has been orphaned.

3.2.2.16 NLME-ADD-NEIGHBOR.request

This optional primitive allows the next higher layer of a Zigbee coordinator or router to request to directly insert another device to the local device's neighbor table.

3.2.2.16.1 Semantics of the Service Primitive

The semantics of this optional primitive are as follows:

```
NLME-ADD-NEIGHBOR.request       {
                                DeviceAddress,
                                CapabilityInformation,
                                InterfaceIndex
                                }
```

Table 3-24 specifies the parameters for the NLME-ADD-NEIGHBOR.request primitive.

Table 3-24. NLME-ADD-NEIGHBOR.request Parameters

Name	Type	Valid Range	Description
DeviceAddress	64-bit IEEE address	Any 64-bit IEEE address	The IEEE address of the device to be directly joined.
CapabilityInformation	Bitmap	See Table 3-67.	The operating capabilities of the device being directly joined.

Name	Type	Valid Range	Description
InterfaceIndex	Integer	0 – 31	This is an index into the MAC Interface Table indicating what interface the neighbor or child is bound to. The neighbor table entry SHALL be set with the value of the MAC Interface Index passed to this primitive

3.2.2.16.2 When Generated

The next higher layer of a Zigbee coordinator or router generates this primitive to add a new device directly to its neighbor table. This process is completed without any over the air transmissions.

3.2.2.16.3 Effect on Receipt

On receipt of this primitive, the NLME will attempt to add the device specified by the DeviceAddress parameter to its neighbor table. The CapabilityInformation parameter will contain a description of the device being joined. The alternate PAN coordinator bit is set to 0 in devices implementing this specification. The device type bit is set to 1 if the device is a Zigbee router, or to 0 if it is an end device. The power source bit is set to 1 if the device is receiving power from the alternating current mains or to 0 otherwise. The receiver on when idle bit is set to 1 if the device does not disable its receiver during idle periods, or to 0 otherwise. The security capability bit is set to 1 if the device is capable of secure operation, or to 0 otherwise.

If the NLME successfully adds the device to its neighbor table, the NLME issues the NLME-ADD-NEIGHBOR.confirm primitive with a status of SUCCESS. If the NLME finds that the requested device is already present in its neighbor tables, the NLME issues the NLME-ADD-NEIGHBOR.confirm primitive with a status of ALREADY_PRESENT. If no capacity is available to add a new device to the device list, the NLME issues the NLME-ADD-NEIGHBOR.confirm primitive with a status of NEIGHBOR_TABLE_FULL.

3.2.2.17 NLME-ADD-NEIGHBOR.confirm

This primitive allows the next higher layer of a Zigbee coordinator or router to be notified of the results of its request to directly add another device to its neighbor table.

3.2.2.17.1 Semantics of the Service Primitive

The semantics of this primitive are as follows:

```
NLME-ADD-NEIGHBOR.confirm       {
                                Status,
                                DeviceAddress
                        }
```

Table 3-25 specifies the parameters for the NLME-ADD-NEIGHBOR.confirm primitive.

Table 3-25. NLME-ADD-NEIGHBOR.confirm Parameters

Name	Type	Valid Range	Description
Status	Status	SUCCESS, ALREADY_PRESENT, NEIGHBOR_TABLE_FULL	The status of the corresponding request.

Name	Type	Valid Range	Description
DeviceAddress	64-bit IEEE address	Any 64-bit IEEE address	The 64-bit IEEE address in the request to which this is a confirmation.

3.2.2.17.2 When Generated

This primitive is generated by the initiating NLME and issued to its next higher layer in response to an NLME-ADD-NEIGHBOR.request primitive. If the request was successful, the Status parameter indicates a successful join attempt. Otherwise, the Status parameter indicates an error code of ALREADY_PRESENT or NEIGHBOR_TABLE_FULL. The reasons for these status values are fully described in section 3.2.2.16.3.

3.2.2.17.3 Effect on Receipt

On receipt of this primitive, the next higher layer of the initiating device is notified of the results of its request to directly join another device to a network.

3.2.2.18 NLME-LEAVE.request

This primitive allows the next higher layer to request that it or another device leaves the network.

3.2.2.18.1 Semantics of the Service Primitive

The semantics of this primitive are as follows:

```
NLME-LEAVE.request          {
                            DeviceAddress,
                            RemoveChildren,
                            Rejoin
                            }
```

Table 3-26 specifies the parameters for the NLME-LEAVE.request primitive.

Table 3-26. NLME-LEAVE.request Parameters

Name	Type	Valid Range	Description
DeviceAddress	Device address	Any 64-bit IEEE address	The 64-bit IEEE address of the entity to be removed from the network or NULL if the device removes itself from the network.
RemoveChildren	Boolean	FALSE	This parameter SHALL be set to FALSE for all NLME-Leave messages.
Rejoin	Boolean	TRUE or FALSE	This parameter has a value of TRUE if the device being asked to leave from the current parent is requested to rejoin the network. Otherwise, the parameter has a value of FALSE. Note that the Rejoin parameter is set by the application so cannot be relied upon by the networking layer to indicate whether a Join or Rejoin request will be accepted in the future.

3.2.2.18.2 When Generated

The next higher layer of a device generates this primitive to request to leave the network. The next higher layer of a Zigbee coordinator or router MAY also generate this primitive to remove a device from the network.

3.2.2.18.3 Effect on Receipt

On receipt of this primitive by the NLME of a device that is not currently joined to a network, the NLME issues the NLME-LEAVE.confirm primitive with a status of INV_REQUESTTYPE. On receipt of this primitive by the NLME of a device that is currently joined to a network, with the DeviceAddress parameter equal to the local device's IEEE address or NULL, the NLME will remove itself from the network using the procedure described in section 3.6.1.11.1, and the value of the Rejoin parameter SHALL be copied into the Network Leave command frame that is generated. If the Rejoin parameter is set to TRUE, no further action is taken. If the Rejoin parameter is set to FALSE the NLME will then clear its routing table and route discovery table and issue an MLME-RESET.request primitive to the MAC sub-layer. The NLME will also set the relationship field of the neighbor table entry corresponding to its former parent to 0x03, indicating no relationship. If the NLME-LEAVE.request primitive is received with the DeviceAddress parameter equal to NULL and the RemoveChildren parameter equal to TRUE, then the NLME will attempt to remove the device's children, as described in section 3.6.1.11.3.

On receipt of this primitive by a Zigbee coordinator or Zigbee router and with the DeviceAddress parameter not equal to NULL and not equal to the local device's IEEE address, the NLME determines whether the specified device is in the Neighbor Table and the device type is 0x02 (Zigbee End device). If the requested device does not exist or the device type is not 0x02, the NLME issues the NLME-LEAVE.confirm primitive with a status of UNKNOWN_DEVICE. If the requested device exists, the NLME will attempt to remove it from the network using the procedure described in section 3.6.1.11.3. If the RemoveChildren parameter is equal to TRUE then the device will be requested to remove its children as well. Following the removal, the NLME will issue the NLME-LEAVE.confirm primitive with the DeviceAddress equal to the 64-bit IEEE address of the removed device and the Status parameter equal to the status delivered by the MCPS-DATA.confirm primitive. If the relationship field of the neighbor table entry corresponding to the leaving device has a value of 0x01 then it will be changed to 0x04 indicating previous child. If it has a value of 0x05, indicating that the child has not yet authenticated, it will be changed to 0x03, indicating no relationship.

3.2.2.19 NLME-LEAVE.indication

This primitive allows the next higher layer of a Zigbee device to be notified if that Zigbee device has left the network or if a neighboring device has left the network.

3.2.2.19.1 Semantics of the Service Primitive

The semantics of this primitive are as follows:

```
NLME-LEAVE.indication       {
                            DeviceAddress,
                            Rejoin
                            }
```

Table 3-27 specifies the parameters for the NLME-LEAVE.indication primitive.

Table 3-27. NLME-LEAVE.indication Parameters

Name	Type	Valid Range	Description
DeviceAddress	64-bit IEEE address	Any 64-bit IEEE address	The 64-bit IEEE address of an entity that has removed itself from the network or NULL in the case that the device issuing the primitive has been removed from the network by its parent.

Name	Type	Valid Range	Description
Rejoin	Boolean	TRUE or FALSE	This parameter has a value of TRUE if the device being asked to leave the current parent is requested to rejoin the network. Otherwise, this parameter has a value of FALSE.

3.2.2.19.2 When Generated

This primitive is generated by the NLME of a Zigbee coordinator or Zigbee router and issued to its next higher layer on receipt of a broadcast leave command pertaining to a device on its PAN. It is also generated by the NLME of a Zigbee router or end device and issued to its next higher layer to indicate that it has been successfully removed from the network by its associated router or Zigbee coordinator.

3.2.2.19.3 Effect on Receipt

On receipt of this primitive, the next higher layer of a Zigbee coordinator or Zigbee router is notified that a device, formerly on the network, has left. The primitive can also inform the next higher layer of a Zigbee router or end device that it has been removed from the network by its associated Zigbee router or Zigbee coordinator parent. In this case, the value of the Rejoin parameter indicates to the next higher layer whether the peer entity on the parent device wishes the device that has been removed to rejoin the same network.

When the local device receives a NLME-LEAVE.indication with Rejoin set to FALSE it SHALL remove any persistent data within the stack related to the leaving device.

When the higher layer is notified of an NLME-LEAVE.indication with Rejoin set to TRUE, it is recommended that no action be taken to remove application information stored about the device (such as bindings).

3.2.2.20 NLME-LEAVE.confirm

This primitive allows the next higher layer of the initiating device to be notified of the results of its request for itself or another device to leave the network.

3.2.2.20.1 Semantics of the Service Primitive

The semantics of this primitive are as follows:

```
NLME-LEAVE.confirm          {
                            Status,
                            DeviceAddress
                            }
```

Table 3-28 specifies the parameters for the NLME-LEAVE.confirm primitive.

Table 3-28. NLME-LEAVE.confirm Parameters

Name	Type	Valid Range	Description
Status	Status	SUCCESS, INV_REQUESTTYPE, UNKNOWN_DEVICE or any status returned by the MCPS-DATA.confirm primitive	The status of the corresponding request.

Name	Type	Valid Range	Description
DeviceAddress	64-bit IEEE address	Any 64-bit, IEEE address	The 64-bit IEEE address in the request to which this is a confirmation or null if the device requested to remove itself from the network.

3.2.2.20.2 When Generated

This primitive is generated by the initiating NLME and issued to its next higher layer in response to an NLME-LEAVE.request primitive. If the request was successful, the Status parameter indicates a successful leave attempt. Otherwise, the Status parameter indicates an error code of INV_REQUESTTYPE, UNKNOWN_DEVICE or a status delivered by the MCPS-DATA.confirm primitive. The reasons for these status values are fully described in section 3.2.2.18.3.

3.2.2.20.3 Effect on Receipt

On receipt of this primitive, the next higher layer of the initiating device is notified of the results of its request for itself or a child device to leave the network.

3.2.2.21 NLME-RESET.request

This primitive allows the next higher layer to request the NWK layer to perform a reset operation.

3.2.2.21.1 Semantics of the Service Primitive

The semantics of this primitive are as follows:

```
NLME-RESET.request          {
                            WarmStart
                            }
```

Table 3-29 specifies the parameters for this primitive.

Table 3-29. NLME-RESET.request Parameters

Name	Type	Valid Range	Description
WarmStart	Boolean	TRUE or FALSE	This parameter has a value of FALSE if the request is expected reset all stack values to their initial default values. If this value is TRUE, the device is expected to resume operations according to the NIB settings prior to the call.

3.2.2.21.2 When Generated

This primitive is generated by the next higher layer and issued to its NLME to request the NWK layer be reset to its initial condition, or that it resume operations according to its current NIB values prior to this primitive being issued.

3.2.2.21.3 Effect on Receipt

On receipt of this primitive, where the WarmStart parameter has a value of FALSE, the NLME issues the MLME-RESET.request primitive to each MAC sub-layer with an entry in the nwkMacInterfaceTable with the SetDefaultPIB parameter set to TRUE. On receipt of the corresponding MLME-RESET.confirm primitive, the NWK layer resets itself by clearing all internal variables, routing table and route discovery table entries and by setting all NIB attributes to their default values. Once the NWK layer is reset, the NLME issues the NLME-RESET.confirm with the Status parameter set to SUCCESS if all the MAC sub-layers were successfully reset or DISABLE_TRX_FAILURE otherwise.

On receipt of this primitive where WarmStart is set to TRUE, the network layer SHOULD NOT modify any NIB values, but rather SHOULD resume normal network operations and consider itself part of the network specified in the NIB. Routing table values and neighbor table values SHOULD be cleared. The method by which the network and MAC layers attributes are pre-loaded is left to the implementer.

If this primitive is issued to the NLME of a device that is currently joined to a network, any required leave attempts using the NLME-LEAVE.request primitive SHOULD be made *a priori* at the discretion of the next higher layer.

3.2.2.22 NLME-RESET.confirm

This primitive allows the next higher layer of the initiating device to be notified of the results of the request to reset the NWK layer.

3.2.2.22.1 Semantics of the Service Primitive

The semantics of this primitive are as follows:

```
NLME-RESET.confirm          {
                            Status
                            }
```

Table 2-30 specifies the parameters for this primitive.

Table 3-30. NLME-RESET.confirm Parameters

Name	Type	Valid Range	Description
Status	Status	SUCCESS, DISABLE_TRX_FAILURE	Refer to section 3.2.2.22.2.

3.2.2.22.2 When Generated

This primitive is generated by the NLME and issued to its next higher layer in response to an NLME-RESET.request primitive. If the request was successful for all MAC sub-layers in the nwkMacInterfaceTable, the Status parameter will have a value of SUCCESS. Otherwise, the Status parameter will indicate an error code of DISABLE_TRX_FAILURE. The reasons for these status values are fully described in section 3.2.2.21.3.

3.2.2.22.3 Effect on Receipt

On receipt of this primitive, the next higher layer is notified of the results of its request to reset the NWK layer.

3.2.2.23 Network Layer Reset Message Sequence Chart

Figure 3-2 illustrates the sequence of messages necessary for resetting the NWK layer.

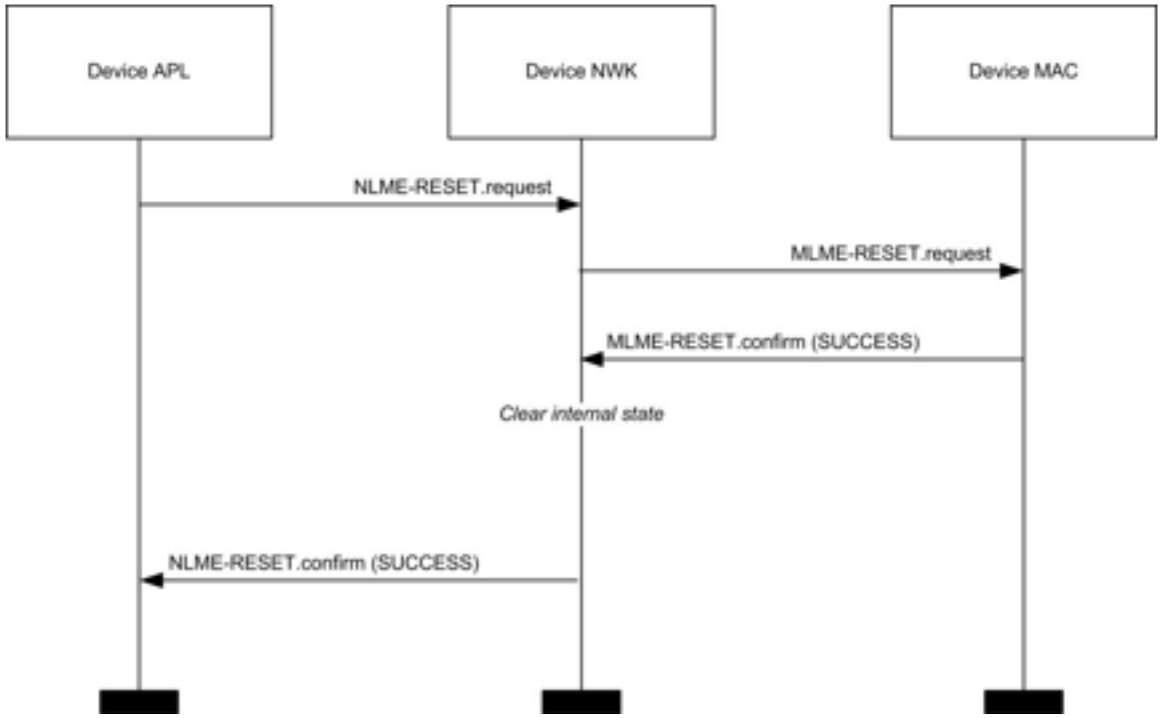


Figure 3-2. Message Sequence Chart for Resetting the Network Layer

3.2.2.24 NLME-SYNC.request

This primitive allows the next higher layer to synchronize or extract data from its Zigbee coordinator or router.

3.2.2.24.1 Semantics of the Service Primitive

This primitive does not have any parameters.

3.2.2.24.2 When Generated

This primitive is generated whenever the next higher layer wishes to achieve synchronization or check for pending data at its Zigbee coordinator or router.

3.2.2.24.3 Effect on Receipt

The NLME issues the MLME-POLL.request primitive to the MAC sub-layer.

If the MLME-POLL.confirm indicates a TRANSACTION_OVERFLOW or a CHANNEL_ACCESS_FAILURE, the device SHALL perform *nwkPerformAdditionalMacDataPollRetries* by issuing additional MLME-POLL.requests until success is returned by the MLME or all retries are exhausted.

Afterwards, an NLME-SYNC.confirm primitive SHALL be issued with the status set to the last result of the MLME-POLL.confirm.

3.2.2.25 NLME-SYNC.confirm

This primitive allows the next higher layer to be notified of the results of its request to synchronize or extract data from its Zigbee coordinator or router.

3.2.2.25.1 Semantics of the Service Primitive

The semantics of this primitive are as follows:

```
NLME-SYNC.confirm          {
                           Status
                           }
```

Table 3-31 specifies the parameters for this primitive.

Table 3-31. NLME-SYNC.confirm Parameters

Name	Type	Valid Range	Description
Status	Status	SUCCESS, SYNC_FAILURE, INVALID_PARAMETER, or any status value returned from the MLME_POLL.confirm primitive (see [B1]).	The result of the request to synchronize with the Zigbee coordinator or router.

3.2.2.25.2 When Generated

This primitive is generated by the initiating NLME and issued to its next higher layer in response to an NLME-SYNC.request primitive. If the request was successful, the Status parameter indicates a successful synchronization attempt. Otherwise, the Status parameter indicates an error code. The reasons for these status values are fully described in section 3.2.2.24.2.

3.2.2.25.3 Effect on Receipt

On receipt of this primitive, the next higher layer is notified of the results of its request to synchronize or extract data from its Zigbee coordinator or router. If the NLME has been successful, the Status parameter will be set to SUCCESS. Otherwise, the Status parameter indicates the error.

3.2.2.26 Message Sequence Charts for Synchronization

illustrates the sequence of messages necessary for a device to successfully synchronize with a Zigbee coordinator or Zigbee router. illustrates the case for a non-beaconing network.

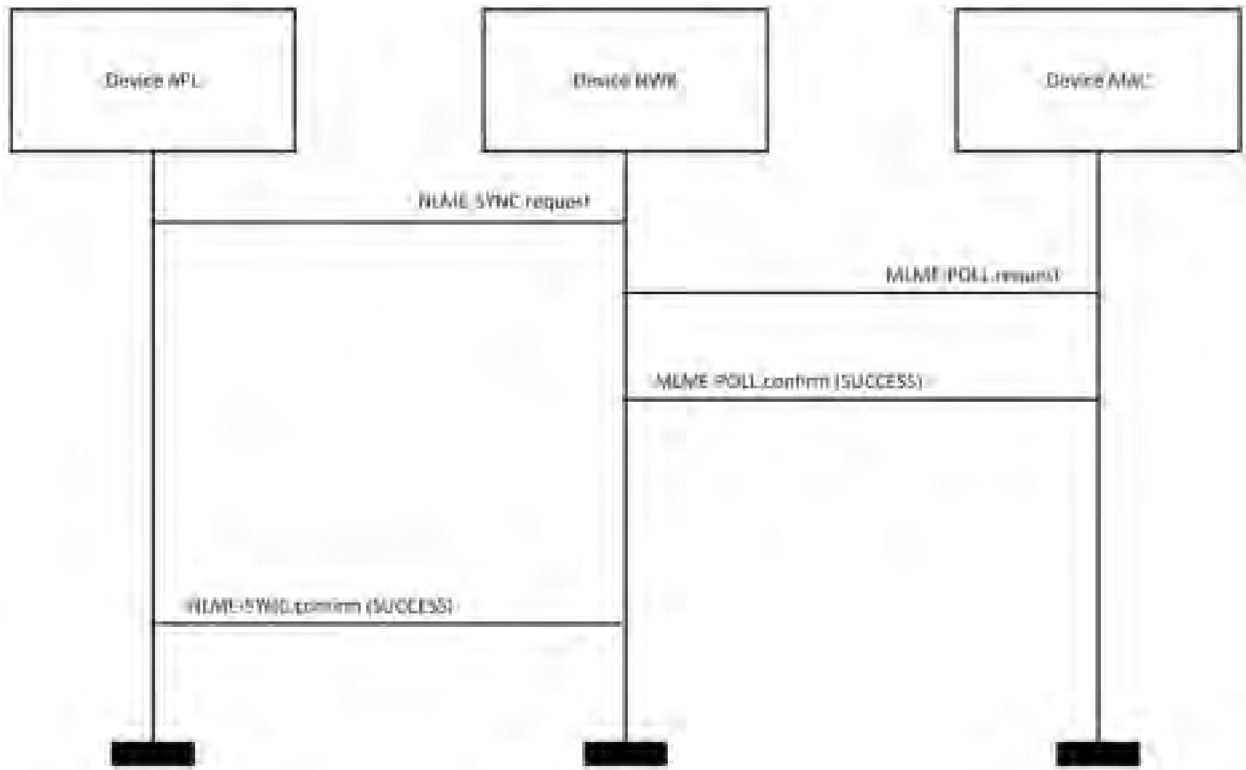


Figure 3-3. Message Sequence Chart for Synchronizing in a Non-Beaconing Network

3.2.2.27 NLME-GET.request

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

3.2.2.27.1 Semantics of the Service Primitive

The semantics of this primitive are as follows:

```
NLME-GET.request          {
                          NIBAttribute
                          }
```

Table 3-32 specifies the parameters for this primitive.

Table 3-32. NLME-GET.request Parameters

Name	Type	Valid Range	Description
NIBAttribute	Integer	See Table 3-62.	The identifier of the NIB attribute to read.

3.2.2.27.2 When Generated

This primitive is generated by the next higher layer and issued to its NLME in order to read an attribute from the NIB.

3.2.2.27.3 Effect on Receipt

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

If the requested NIB attribute is successfully retrieved, the NLME issues the NLME-GET.confirm primitive with a status of SUCCESS and the NIB attribute identifier and value.

3.2.2.28 NLME-GET.confirm

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

3.2.2.28.1 Semantics of the Service Primitive

The semantics of this primitive are as follows:

NLME-GET.confirm	{
	Status,
	NIBAttribute,
	NIBAttributeLength,
	NIBAttributeValue
	}

Table 3-33 specifies the parameters for this primitive.

Table 3-33. NLME-GET.confirm Parameters

Name	Type	Valid Range	Description
Status	Enumeration	SUCCESS or UNSUPPORTED_ATTRIBUTE	The results of the request to read a NIB attribute value.
NIBAttribute	Integer	See Table 3-62.	The identifier of the NIB attribute that was read.
NIBAttributeLength	Integer	0x0000 – 0xffff	The length, in octets, of the attribute value being returned.
NIBAttributeValue	Various	Attribute-specific (see Table 3-62)	The value of the NIB attribute that was read.

3.2.2.28.2 When Generated

This primitive is generated by the NLME and issued to its next higher layer in response to an NLME-GET.request primitive. This primitive returns either a status of SUCCESS, indicating that the request to read a NIB attribute was successful, or an error code of UNSUPPORTED_ATTRIBUTE. The reasons for these status values are fully described in section 3.2.2.27.3.

3.2.2.28.3 Effect on Receipt

On receipt of this primitive, the next higher layer is notified of the results of its request to read a NIB attribute. If the request to read a NIB attribute was successful, the Status parameter will be set to SUCCESS. Otherwise, the Status parameter indicates the error.

3.2.2.29 NLME-SET.request

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

3.2.2.29.1 Semantics of the Service Primitive

The semantics of this primitive are as follows:

```
NLME-SET.request          {
                          NIBAttribute,
                          NIBAttributeLength,
                          NIBAttributeValue
                          }
```

Table 3-34 specifies the parameters for this primitive.

Table 3-34. NLME-SET.request Parameters

Name	Type	Valid Range	Description
NIBAttribute	Integer	See Table 3-62.	The identifier of the NIB attribute to be written.
NIBAttributeLength	Integer	0x0000 – 0xffff	The length, in octets, of the attribute value being set.
NIBAttributeValue	Various	Attribute-specific (see Table 3-62).	The value of the NIB attribute that SHOULD be written.

3.2.2.29.2 When Generated

This primitive is to be generated by the next higher layer and issued to its NLME in order to write the value of an attribute in the NIB.

3.2.2.29.3 Effect on Receipt

On receipt of this primitive the NLME attempts to write the given value to the indicated NIB attribute in its database. If the NIBAttribute parameter specifies an attribute that is not found in the database, the NLME issues the NLME-SET.confirm primitive with a status of UNSUPPORTED_ATTRIBUTE. If the NIBAttributeValue parameter specifies a value that is out of the valid range for the given attribute, the NLME issues the NLME-SET.confirm primitive with a status of INVALID_PARAMETER.

If the requested NIB attribute is successfully written, the NLME issues the NLME-SET.confirm primitive with a status of SUCCESS.

3.2.2.30 NLME-SET.confirm

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

3.2.2.30.1 Semantics of the Service Primitive

The semantics of this primitive are as follows:

```
NLME-SET.confirm          {
                          Status,
                          NIBAttribute
                          }
```

Table 3-35 specifies the parameters for this primitive.

Table 3-35. NLME-SET.confirm Parameters

Name	Type	Valid Range	Description
Status	Enumeration	SUCCESS, INVALID_PARAMETER, or UNSUPPORTED_ATTRIBUTE	The result of the request to write the NIB attribute.
NIBAttribute	Integer	See Table 3-62.	The identifier of the NIB attribute that was written.

3.2.2.30.2 When Generated

This primitive is generated by the NLME and issued to its next higher layer in response to an NLME-SET.request primitive. This primitive returns a status of either SUCCESS, indicating that the requested value was written to the indicated NIB attribute, or an error code of INVALID_PARAMETER or UNSUPPORTED_ATTRIBUTE. The reasons for these status values are fully described in section 3.2.2.29.3.

3.2.2.30.3 Effect on Receipt

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

3.2.2.31 NLME-NWK-STATUS.indication

This primitive allows the next higher layer to be notified of network failures.

3.2.2.31.1 Semantics of the Service Primitive

The semantics of this primitive are as follows:

```
NLME-NWK-STATUS.indication        {
                                  Status,
                                  NetworkAddr
                                  }
```

Table 3-36 specifies the parameters for this primitive.

Table 3-36. NLME-NWK-STATUS.indication Parameters

Name	Type	Valid Range	Description
Status	Status	Any network status code (see Table 3-52).	The error code associated with the failure.
NetworkAddr	Integer	0x0000 – 0xfff7	The 16-bit network address of the device associated with the status information.

3.2.2.31.2 When Generated

This primitive is generated by the NWK layer on a device and passed to the next higher layer when one of the following occurs:

The device has failed to discover or repair a route to the destination given by the NetworkAddr parameter.

The NWK layer on that device has failed to deliver a frame to its end device child with the 16-bit network address given by the NetworkAddr parameter, due to one of the reasons given in Table 3-52.

The NWK layer has received a network status command frame destined for the device. In this case, the values of the NetworkAddr and Status parameters will reflect the values of the destination address and error code fields of the command frame.

3.2.2.31.3 Effect on Receipt

The next higher layer is notified of a failure to communicate with the identified device.

3.2.2.32 NLME-ROUTE-DISCOVERY.request

This primitive allows the next higher layer to initiate route discovery.

3.2.2.32.1 Semantics of the Service Primitive

The semantics of this primitive are as follows:

```
NLME-ROUTE-DISCOVERY.request    {
                                DstAddrMode,
                                DstAddr,
                                Radius,
                                NoRouteCache
                                }
```

Table 3-37 specifies the parameters for this primitive.

Table 3-37. NLME-ROUTE-DISCOVERY.request Parameters

Name	Type	Valid Range	Description
DstAddrMode	Integer	0x00 – 0x02	A parameter specifying the kind of destination address provided. The DstAddrMode parameter MAY take one of the following three values: 0x00 = No destination address 0x01 = Reserved 0x02 = 16-bit network address of an individual device
DstAddr	16-bit network address	Any network address	The destination of the route discovery. If the DstAddrMode parameter has a value of 0x00 then no DstAddr will be supplied. This indicates that the route discovery will be a many-to-one discovery with the device issuing the discovery command as a target. If the DstAddrMode parameter has a value of 0x02, this indicates unicast route discovery. The DstAddr will be the 16-bit network address of a device to be discovered.

Name	Type	Valid Range	Description
Radius	Integer	0x00 – 0xff	This optional parameter describes the number of hops that the route request will travel through the network.
NoRouteCache	Boolean	TRUE or FALSE	In the case where DstAddrMode has a value of zero, indicating many-to-one route advertisement, this flag determines whether the NWK SHOULD establish a route record table. TRUE = no route record table SHOULD be established FALSE = establish a route record table

3.2.2.32.2 When Generated

This primitive is generated by the next higher layer of a Zigbee coordinator or router and issued to its NLME to request the initiation of route discovery.

3.2.2.32.3 Effect on Receipt

On receipt of this primitive by the NLME of a Zigbee end device, the NLME will issue the NLME-ROUTE-DISCOVERY.confirm primitive to the next higher layer with a status value of INV_REQUESTTYPE.

On receipt of this primitive by the NLME with the DstAddrMode parameter not equal to 0x00 and the DstAddr parameter equal to a broadcast address, the NLME will issue the NLME-ROUTE-DISCOVERY.confirm primitive to the next higher layer with a status value of INV_REQUESTTYPE.

In each of the three cases of actual route discovery described above, the NLME will initiate route discovery by attempting to transmit a route discovery command frame using the MCPS-DATA.request primitive of the MAC sub-layer. If a value has been supplied for the optional Radius parameter, that value will be placed in the Radius field of the NWK header of the outgoing frame. If a value has not been supplied then the radius field of the NWK header will be set to twice the value of the *nwkcMaxDepth* attribute of the NIB as would be the case for data frame transmissions. If the MAC sub-layer fails, for any reason, to transmit the route request command frame, the NLME will issue the NLME-ROUTE-DISCOVERY.confirm primitive to the next higher layer with a Status parameter value equal to that returned by the MCPS-DATA.confirm. If the route discovery command frame is sent successfully and if the DstAddrMode parameter has a value of 0x00, indicating many-to-one route advertisement, the NLME will immediately issue the NLME-ROUTE-DISCOVERY.confirm primitive with a value of SUCCESS. Otherwise, the NLME will wait until either a route reply or route record command frame is received or a reactive many-to-one route request command originating in the device identified by DstAddr is received or the route discovery operation times out as described in section 3.6.4.5. If a route reply or route record or matching reactive many-to-one route request command frame is received before the route discovery operation times out, the NLME will issue the NLME-ROUTE-DISCOVERY.confirm primitive to the next higher layer with a status value of SUCCESS. If the operation times out, it will issue the NLME_ROUTE-DISCOVERY.confirm primitive with a Status of ROUTE_ERROR and with a NetworkStatusCode value reflecting the reason for failure as described in Table 3-52.

3.2.2.33 NLME-ROUTE-DISCOVERY.confirm

This primitive informs the next higher layer about the results of an attempt to initiate route discovery.

3.2.2.33.1 Semantics of the Service Primitive

The semantics of this primitive are as follows:

```
NLME-ROUTE-DISCOVERY.confirm            {
                                        Status,
                                        NetworkStatusCode
                                        }
```

Table 3-38 specifies the parameters for the NLME-ROUTE-DISCOVERY.confirm primitive.

Table 3-38. NLME_ROUTE-DISCOVERY.confirm Parameters

Name	Type	Valid Range	Description
Status	Status value	INV_REQUESTTYPE, ROUTE_ERROR, or any status value returned by the MCPS-DATA.confirm primitive.	The status of an attempt to initiate route discovery.
Network-StatusCode	Network status code	See Table 3-52.	In the case where the Status parameter has a value of ROUTE_ERROR, this code gives further information about the kind of error that occurred. Otherwise, it SHOULD be ignored.

3.2.2.33.2 When Generated

This primitive is generated by the NLME and passed to the next higher layer as a result of an attempt to initiate route discovery

3.2.2.33.3 Effect on Receipt

The next higher layer is informed of the status of its attempt to initiate route discovery. Possible values for the Status parameter and the circumstances under which they are generated are described in section 3.2.2.32.3. NLME-SET-INTERFACE.request.

This primitive allows the next higher layer to request that the NWK layer enable or disable an interface in the MAC Interface Table.

3.2.2.34 NLME-SET-INTERFACE.request

This primitive allows the next higher layer to request that the NWK layer enable or disable an interface in the MAC Interface Table.

3.2.2.34.1 Semantics of the Service Primitive

The semantics of this primitive are as follows:

```
NLME-SET-INTERFACE.request          {
                                    InterfaceIndex
                                    State
                                    ChannelToUse
                                    SupportedChannels
                                    RoutersAllowed
                                    DutyCycleWarningThreshold
                                    DutyCycleCriticalThreshold
                                    DutyCycleRegulatedThreshold
                                    InterfaceLinkCostScalar
                                    }
```

Table 3-39 specifies the parameters for the NLME-SET-INTERFACE.request primitive.

Table 3-39. NLME-SET-INTERFACE.request Parameters

Name	Type	Valid Range	Description
InterfaceIndex	Integer	0 – 31	The index of the interface in the Mac Interface Table to set.
State	Boolean	TRUE or FALSE	This enables or disables the interface. TRUE indicates to enable, FALSE indicates to disable.
ChannelToUse	Channel Page Structure	Variable	This indicates a single channel that the interface will be set to. NULL indicates unspecified channel.
SupportedChannels	Channel List Structure	Variable	This indicates the complete set of channels supported by the specified interface.
RoutersAllowed	Boolean	TRUE or FALSE	This indicates whether routers are allowed to join to this device on this interface.
DutyCycleWarningThreshold	Integer	0 – 1024	This enables the Duty Cycle Warning threshold to be set. The integer value set is 10x the required threshold in percent, for example, 1% is set by a value of 10. A value of 0 indicates that there is no limit on Duty Cycle.
DutyCycleCritical-Threshold	Integer	0 – 1024	This enables the Duty Cycle Critical threshold to be set. The integer value set is 10x the required threshold in percent, for example, 1% is set by a value of 10. A value of 0 indicates that there is no limit on Duty Cycle.

Name	Type	Valid Range	Description
DutyCycleRegulatedThreshold	Integer	0 – 1024	This enables the Duty Cycle Regulated threshold to be set. The integer value set is 10x the required threshold in percent, for example, 1% is set by a value of 10. A value of 0 indicates that there is no limit on Duty Cycle.
InterfaceLinkCostScalar	Integer	1 – 36	This is used to scale all of the Link costs on the interface.

3.2.2.34.2 When Generated

This primitive is generated by the next higher layer when it wants to change the interfaces of the NWK layer.

3.2.2.34.3 Effect on Receipt

1. Upon receipt of this primitive the NWK layer SHALL find the corresponding entry in the nwkMacInterfaceTable where the Index value matches the InterfaceIndex parameter passed via this primitive.
    a. If no such entry exists, then the NLME issues an NLME-SET-INTERFACE.confirm primitive with the Status parameter set to INV_REQUESTTYPE, and no more processing SHALL take place.
2. If the State passed to this primitive is set to FALSE,
    a. The NLME SHALL examine the State values of all other interfaces in the nwkMacInterface table.
        i. If all other interfaces are set to FALSE then the NLME SHALL issue a NLME-SET-INTERFACE.confirm with a Status of INV_REQUESTTYPE and no further processing SHALL be done.
        ii. If the MAC Interface Index in the entry is the same as the InterfaceIndex passed to this primitive, disable the entry from the nwkMacInterfaceTable.
        iii. Disabling the interface is done as follows:
            1. Issue an MLME-RX-ENABLE.request with the following parameters:
                a. DeferPermit SHALL be set to FALSE
                b. RxOnTime is set to 0.
                c. RxOnDuration is set to 0.
            2. Wait until an MLME-RX-ENABLE.confirm has been issued.
            3. For each MSDU in the MAC queue, issue an MCPS-PURGE.request.
            4. Issue an NLME-SET-INTERFACE.confirm with the Status set to the Status returned by the MLME-RX-ENABLE.confirm primitive.
            5. No more processing SHALL be done.
3. If the State passed to this primitive is set to TRUE,
    a. The NWK layer SHALL verify that the Channel value requested for the corresponding index is valid for the interface entry.
        i. If not the NLME issues an NLME-SET-INTERFACE.confirm primitive with the Status of INV_REQUESTTYPE, and no more processing SHALL take place.
    b. The NLME SHALL examine the ChannelToUse parameter and validate that a single channel is specified. It SHALL then verify that the ChannelToUse corresponds to a channel indicated in the SupportedChannels parameter.

i. If the tests in 3.b do not pass, then processing SHALL fail and the NLME issues an NLME-SET-INTERFACE.confirm primitive with the Status of INV_REQUESTTYPE and no further processing SHALL take place.

c. If the tests in 3.b do pass, the NLME issues a NLME-SET-INTEFRACE.comfirm primitive with the status parameter set to Success. It SHALL then set ChannelInUse of the interface entry in the nwkMacInterfaceTable to the Channel~~In~~ToUse value passed into this primitive. The NLME SHALL set the State value according to the State value passed into this primitive. The NLME SHALL set the RoutersAllowed of the interface entry to the RoutersAllowed parameter passed to this primitive.

3.2.2.35 NLME-SET-INTERFACE.confirm

This primitive allows the NLME to notify the next higher layer of the result of an NLME-SET-INTERFACE.request.

3.2.2.35.1 Semantics of the Service Primitive

The semantics of this primitive are as follows:

```
NLME-SET-INTERFACE.confirm      {
                                Status
                                }
```

Table 3-40 specifies the parameters for the NLME-SET-INTERFACE.confirm primitive.

Table 3-40. NLME-SET-INTERFACE.confirm parameters

Name	Type	Valid Range	Description
Status	Status value	SUCCESS or INV_REQUESTTYPE	The result of a previous NLME-SET-INTERFACE.request call.

3.2.2.35.2 When Generated

This primitive is generated by the NLME when it wants to notify the next higher layer of the result of an NLME-SET-INTERFACE.request.

3.2.2.35.3 Effect on Receipt

The next higher layer will be informed about the result to change the interfaces of the NWK layer.

3.2.2.36 NLME-GET-INTERFACE.request

This primitive allows the next higher layer to request information from the NWK layer about an interface in the MAC Interface Table.

3.2.2.36.1 Semantics of the Service Primitive

The semantics of this primitive are as follows:

```
NLME-GET-INTERFACE.request      {
                                InterfaceIndex
                                }
```

Table 3-41 specifies the parameters for the NLME-GET-INTERFACE.request primitive.

Table 3-41. NLME-GET-INTERFACE.request primitive

Name	Type	Valid Range	Description
InterfaceIndex	Integer	0 – 31	The index of the interface in the Mac Interface Table to retrieve.

3.2.2.36.2 When Generated

This primitive is generated by the next higher layer when it wants to retrieve information about an interface from the NWK layer.

3.2.2.36.3 Effect on Receipt

Upon receipt of this primitive the NWK layer SHALL find the corresponding entry in the *nwkMacInterfaceTable* where the Index value matches the InterfaceIndex parameter passed via this primitive. If no such entry exists, then the NLME issues an NLME-GET-INTERFACE.confirm primitive with the Status parameter set to INV_REQUESTTYPE, and no more processing SHALL be done.

The NLME SHALL retrieve the parameters from the entry of the nwkMacInterfaceTable and issue the NLME-GET-INTERFACE.confirm primitive.

3.2.2.37 NLME-GET-INTERFACE.confirm

This primitive allows the NLME to notify the next higher layer of the result of a previous NLME-GET-INTEFRACE.request primitive. The values of MacTxUcastTotal, MacTxUcastRetries, MacTxUcastFailures, and MacRxUcast for the interface SHALL be reset to zero upon successful generation of NLME-GET-INTERFACE.confirm. Therefore subsequent NLME-GET-INTERFACE.request operations SHALL return the set of values since the last NLME-GET-INTERFACE.request

3.2.2.37.1 Semantics of the Service Primitive

The semantics of this primitive are as follows:

```
NLME-GET-INTERFACE.confirm          {
                                    InterfaceIndex
                                    Status
                                    State
                                    ChannelInUse
                                    SupportedChannels
                                    RoutersAllowed
                                    PowerNegotiationSupported
                                    DutyCycleWarningThreshold
                                    DutyCycleCriticalThreshold
                                    DutyCycleRegulatedThreshold
                                    MacRxUcast
                                    MacTxUcastRetries
                                    MacTxUcastFailures
                                    MacTxUcastTotal
                                    InterfaceLinkCostScalar
                                    }
```

Table 3-42 specifies the parameters for the NLME-GET-INTERFACE.confirm primitive.

Table 3-42. NLME-GET-INTERFACE.confirm parameters

Name	Type	Valid Range	Description
InterfaceIndex	Integer	0 – 31	The index of the interface in the Mac Interface Table to set.
Status	Status value	SUCCESS or INV_REQUESTTYPE	The result of a previous NLME-GET-INTERFACE.request call.
State	Boolean	TRUE or FALSE	This returns the state of the interface. TRUE indicates to enable, FALSE indicates to disable.
ChannelInUse	Channel Page Structure	Variable	This indicates a single channel that the interface is currently set to. This only applies when the State is set to TRUE.
SupportedChannels	Channel List Structure	Variable	This indicates the complete set of channels supported by the specified interface.
RoutersAllowed	Boolean	TRUE or FALSE	A Boolean indicating whether or not routers are allowed to join to this device on this interface.
PowerNegotiationSupported	Boolean	TRUE or FALSE	Indicates whether this interface supports dynamic power control and negotiation to reduce power output on a per neighbor basis.
DutyCycleWarningThreshold	Integer	0 – 10000	This indicates the value currently set for the Duty Cycle Warning threshold. The integer value set is 100x the required threshold in percent, for example, 1% is set by a value of 100. A value of 0 indicates that there is no limit on Duty Cycle.
DutyCycleCriticalThreshold	Integer	0 – 10000	This indicates the value currently set for the Duty Cycle Critical threshold. The integer value set is 100x the required threshold in percent, for example, 1% is set by a value of 100. A value of 0 indicates that there is no limit on Duty Cycle.
DutyCycleRegulatedThreshold	Integer	0 – 10000	This indicates the value currently set for the Duty Cycle Regulated threshold. The integer value set is 100x the required threshold in percent, for example, 1% is set by a value of 100. A value of 0 indicates that there is no limit on Duty Cycle.

Name	Type	Valid Range	Description
MacRxUcast	Integer	0 – 0xffff	Total received packets, counting retried messages but not counting ACKs or CRC failures.
MacTxUcastRetries	Integer	0 – 0xffff	Total number of Mac Retries regardless of whether the transactions resulted in success or failure
MacTxUcastFailures	Integer	0 – 0xffff	Total number of failed Tx Transactions. So if the Mac sent a single packet, and it is retried 4 times without ACK, that counts as 1 failure
MacTxUcastTotal	Integer	0 – 0xffff	Total number of Mac Tx Transactions to attempt to send a message (but not counting retries)
InterfaceLinkCostScalar	Integer	1 – 36	This is used to scale all of the Link costs on the interface.

3.2.2.37.2 When Generated

This primitive is generated by the NLME to return the result of a previous NLME-GET-INTERFACE.request primitive.

3.2.2.37.3 Effect on Receipt

The higher level application MAY use the information to inform its operation.

3.2.2.38 NLME-DUTY-CYCLE-MODE.indication

3.2.2.38.1 Semantics of this primitive

The semantics of this primitive are as follows:

```
NLME-DUTY-CYCLE-MODE.indication          {
                                         InterfaceIndex
                                         Status
                                         }
```

Table 3-43 specifies the parameters for the NLME-DUTY-CYCLE-MODE.indication primitive.

Table 3-43. NLME-DUTY-CYCLE-MODE.indication parameters

Name	Type	Valid Range	Description
InterfaceIndex	Integer	0 – 31	The index of the interface in the Mac Interface Table to set.

Name	Type	Valid Range	Description
Status	Enumeration	Any valid status returned from the MLME-DUTY-CYCLE-MODE.indication primitive.	The duty cycle mode that the device is currently operating in.

3.2.2.38.2 When Generated

This primitive is generated by the NLME with the new duty cycle status every time MLME-DUTY-CYCLE-MODE.indication changes on any MAC interface.

3.2.2.38.3 Effect on Receipt

The higher level application MAY use the information to inform its operation.

3.2.2.39 NLME-END-DEVICE-NEGOTIATE.request

This primitive allows the next higher layer of an End Device to negotiate or re-negotiate the parameters with its parent. This includes elements like the end device timeout or enabling other features that require both parent and end device to synchronize.

3.2.2.39.1 Semantics of this primitive

The semantics of this primitive are as follows:

```
NLME-END-DEVICE-NEGOTIATE.request       {
                                        EndDeviceTLVs
                                        }
```

Table 3-44 specifies the parameters for the NLME- END-DEVICE-NEGOTIATE.request primitive.

Table 3-44. Parameters of the NLME-END-DEVICE-NEGOTIATE.request primitive

Name	Type	Valid Range	Description
EndDeviceTLVs	TLVs	Variable	This may contain a set of TLVs that indicate information that is being provided to the parent or features that are being requested. This parameter may also be omitted.

3.2.2.39.2 When Generated

This primitive is generated whenever the next higher layer wishes to change the timeout, provide information to the parent, or request features be enabled by the parent.

If the application layer wishes to change the End Device Timeout it will first set the NIB value *nwkEndDeviceTimeout* via NLME-SET.req and then call this primitive. If the application layer does not wish to change the timeout value then it will leave the NIB value alone. The Application layer may still initiate this primitive to transmit other TLVs to the parent.

3.2.2.39.3 Effect on Receipt

On receipt of this primitive the stack will generate an End Device Timeout Request command frame with the value of the nwkEndDeviceTimeout from the NIB. If EndDeviceTLVs have been specified it will include this in the network command frame.

To obtain the result of the End Device Timeout Request and report the result to the application, the stack SHALL issue one or more NLME-SYNC.request primitives to poll for the result.

The result of the operation is returned to the application via the NLME-END-DEVICE-NEGOTIATE.confirm.

3.2.2.40 NLME-END-DEVICE-NEGOTIATE.confirm

This primitive allows the next higher layer to be notified of the results of its request to negotiate or re-negotiate the parameters with its parent.

3.2.2.40.1 Semantics of this primitive

The semantics of this primitive are as follows:

```
NLME-END-DEVICE-NEGOTIATE.confirm        {
                                         Status
                                         }
```

Table 3-45 specifies the parameters for the NLME- END-DEVICE-NEGOTIATE.confirm primitive.

Table 3-45. Parameters of the NLME-END-DEVICE-NEGOTIATE.confirm primitive

Name	Type	Valid Range	Description
Status	Enumeration	Variable	This contains the result of the attempt to negotiate the parameters of the end device to parent connection.

3.2.2.40.2 When Generated

This is generated by the Network layer to inform the application layer about a recent attempt to negotiate the parameters of the end device to parent connection, such as the timeout.

3.2.2.40.3 Effect on Receipt

When the status is SUCCESS, the application usually does not need to take any action. When the status indicates a failure, the application may wish to retry the operation and vary the parameters of the negotiation attempt.

Table 3-46. NIB Attributes

Attribute	Id	Type	Read Only	Range	Description	Default
nwkMaxRejoinParentAttempts	0xC3	Integer	No	0 – 255	The maximum number of attempts to rejoin to parent devices for the current network.	3
nwkEndDeviceTimeout	0xC4	Integer	No	0 – 14	The enumerated timeout value that the local end device stack will use when negotiating its end device timeout. This value is converted to seconds by referencing Table 3-52 Requested Timeout Enumerated Values. This value is ignored for a router or coordinator device. This value is only used when the local device is an end device.	

3.3 Frame Formats

This section specifies the format of the NWK frame (NPDU). Each NWK frame consists of the following basic components:

- A NWK header, which comprises frame control, addressing and sequencing information
- A NWK payload, of variable length, which contains information specific to the frame type

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

3.3.1 General NPDU Frame Format

The NWK frame format is composed of a NWK header and a NWK payload. The fields of the NWK header appear in a fixed order. The NWK frame SHALL be formatted as illustrated in Figure 3-4.

Octets: 2	2	2	1	1	0/8	0/8		Variable	Variable
Frame control	Destination address	Source address	Radius	Sequence number	Destination IEEE Address	Source IEEE Address		Source route subframe	Frame payload
NWK Header									Payload

Figure 3-4. General NWK Frame Format

3.3.1.1 Frame Control Field

The frame control field is 16 bits in length and contains information defining the frame type, addressing and sequencing fields and other control flags. The frame control field SHALL be formatted as illustrated in Figure 3-5.

Bits: 0-1	2-5	6-7	8	9	10	11	12	13	14-15
Frame type	Protocol version	Discover route	Deprecated (Multicast flag)	Security	Source Route	Destination IEEE Address	Source IEEE Address	End Device Initiator	Reserved

Figure 3-5. Frame Control Field

Table 3-47 shows the allowable frame control sub-field configurations for NWK data frames. Note that all frames listed below will have a frame type sub-field equal to 00 indicating data and a protocol version sub-field reflecting the version of the Zigbee specification implemented.

Table 3-47. Allowable Frame Control Sub-Field Configurations

Data Transmission Method	Discover Route	Multicast	Security	Destination IEEE Address	Source IEEE Address
Unicast	00 or 01	0	0 or 1	0 or 1	0 or 1
Broadcast	00	0	0 or 1	0	0 or 1
Source routed	00	0	0 or 1	0 or 1	0 or 1

3.3.1.1.1 Frame Type Sub-Field

The frame type sub-field is 2 bits in length and SHALL be set to one of the non-reserved values listed in Table 3-48.

Table 3-48. Values of the Frame Type Sub-Field

Frame Type Value b_1 b_0	Frame Type Name
00	Data
01	NWK command
10	Reserved
11	Inter-PAN

3.3.1.1.2 Protocol Version Sub-Field

The protocol version sub-field is 4 bits in length and SHALL be set to a number reflecting the Zigbee NWK protocol version in use. The protocol version in use on a particular device SHALL be made available as the value of the NWK constant *nwkcProtocolVersion*.

3.3.1.1.3 Discover Route Sub-Field

The discover route sub-field MAY be used to control route discovery operations for the transit of this frame (see section 3.6.4.5).

Table 3-49. Values of the Discover Route Sub-Field

Discover Route Field Value	Field Meaning
0x00	Suppress route discovery
0x01	Enable route discovery
0x02, 0x03	Reserved

For NWK layer command frames, the discover route sub-field SHALL be set to 0x00 indicating suppression of route discovery.

3.3.1.1.4 Security Sub-Field

The security sub-field SHALL have a value of 1 if, and only if, the frame is to have NWK security operations enabled. If security for this frame is implemented at another layer or disabled entirely, it SHALL have a value of 0.

3.3.1.1.5 Source Route Sub-Field

The source route sub-field SHALL have a value of 1 if and only if a source route subframe is present in the NWK header. If the source route subframe is not present, the source route sub-field SHALL have a value of 0.

3.3.1.1.6 Destination IEEE Address Sub-Field

The destination IEEE address sub-field SHALL have a value of 1 if, and only if, the NWK header is to include the full IEEE address of the destination.

3.3.1.1.7 Source IEEE Address Sub-Field

The source IEEE address sub-field SHALL have a value of 1 if, and only if, the NWK header is to include the full IEEE address of the source device.

3.3.1.1.8 End Device Initiator

If the source of the message is an end device and the *nwkParentInformation* field of the NIB is a value other than 0, then this sub-field SHALL be set to 1. Otherwise this sub-field SHALL be set to 0. After validating the source (see section 3.6.2.2), a router parent device SHALL clear this field when relaying a message sent by one of its end device children.

3.3.1.2 Destination Address Field

The destination address field SHALL always be present and SHALL be 2 octets in length. If the multicast flag sub-field of the frame control field has the value 0, the destination address field SHALL hold the 16-bit network address of the destination device or a broadcast address (see Table 3-76). Note that the network address of a device SHALL be set to the value of the *macShortAddress* attribute of the MAC PIB.

3.3.1.3 Source Address Field

The source address field SHALL always be present. It SHALL always be 2 octets in length and SHALL hold the network address of the source device of the frame. Note that the network address of a device SHALL be set to value of the *macShortAddress* attribute of the MAC PIB.

3.3.1.4 Radius Field

The radius field SHALL always be present. It will be 1 octet in length and specifies the range of a radius-limited transmission. The field SHALL be decremented by 1 by each receiving device.

3.3.1.5 Sequence Number Field

The sequence number field is present in every frame and is 1 octet in length. The sequence number value SHALL be incremented by 1 with each new frame transmitted. The values of the source address and sequence number fields of a frame, taken as a pair, MAY be used to uniquely identify a frame within the constraints imposed by the sequence number's one-octet range. For more details on the use of the sequence number field, see section 3.6.2.

3.3.1.6 Destination IEEE Address Field

The destination IEEE address field, if present, contains the 64-bit IEEE address corresponding to the 16-bit network address contained in the destination address field of the NWK header. Upon receipt of a frame containing a 64-bit IEEE address, the contents of the *nwkAddressMap* and neighbor table SHOULD be checked for consistency, and updated if necessary. Section 3.6.1.10.2 describes the actions to take in detecting address conflicts. If the 16-bit network address is a broadcast or multicast address then the destination IEEE address field SHALL NOT be present.

3.3.1.7 Source IEEE Address Field

The source IEEE address field, if present, contains the 64-bit IEEE address corresponding to the 16-bit network address contained in the source address field of the NWK header. Upon receipt of a frame containing a 64-bit IEEE address, the contents of the *nwkAddressMap* and Neighbor Table SHOULD be checked for consistency, and updated if necessary. Section 3.6.1.10.2 describes the actions to take in detecting address conflicts.

3.3.1.8 Source Route Subframe Field

The source route subframe field SHALL only be present if the source route sub-field of the frame control field has a value of 1. It is divided into three sub-fields as illustrated in Figure 3-6.

Octets: 1	**1**	**Variable**
Relay count	Relay index	Relay list

Figure 3-6. Source Route Subframe Format

3.3.1.8.1 Relay Count Sub-Field

The relay count sub-field indicates the number of relays contained in the relay list sub-field of the source route sub-frame.

3.3.1.8.2 Relay Index

The relay index sub-field indicates the index of the next relay in the relay list sub-field to which the packet will be transmitted. This field is initialized to 1 less than the relay count by the originator of the packet, and is decremented by 1 by each receiving relay.

3.3.1.8.3 Relay List Sub-Field

The relay list sub-field SHALL contain the list of relay addresses. The relay closest to the destination SHALL be listed first. The relay closest to the originator SHALL be listed last.

3.3.1.8.4 Frame Payload Field

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

3.3.2 Format of Individual Frame Types

There are two defined NWK frame types: data and NWK command. Each of these frame types is discussed in the following sections.

3.3.2.1 Data Frame Format

The data frame SHALL be formatted as illustrated in Figure 3-7.

Octets: 2	**Variable**	**Variable**
Frame control	Routing fields	Data payload
NWK header		NWK payload

Figure 3-7. Data Frame Format

The order of the fields of the data frame SHALL conform to the order of the general NWK frame format as illustrated in Figure 3-4.

3.3.2.1.1 Data Frame NWK Header Field

The data frame NWK header field SHALL contain the frame control field and an appropriate combination of routing fields as required.

In the frame control field, the frame type sub-field SHALL contain the value that indicates a data frame, as shown in Table 3-48. All other sub-fields shall be set according to the intended use of the data frame.

The routing fields SHALL contain an appropriate combination of address and broadcast fields, depending on the settings in the frame control field (see Figure 3-5).

3.3.2.1.2 Data Payload Field

The data frame data payload field SHALL contain the sequence of octets that the next higher layer has requested the NWK layer to transmit.

3.3.2.2 NWK Command Frame Format

The NWK command frame SHALL be formatted as illustrated in Figure 3-8.

Octets: 2	Variable	1	Variable
Frame control	Routing fields	NWK command identifier	NWK command payload
NWK header		NWK payload	

Figure 3-8. NWK Command Frame Format

The order of the fields of the NWK command frame SHALL conform to the order of the general NWK frame as illustrated in .

3.3.2.2.1 NWK Command Frame NWK Header Field

The NWK header field of a NWK command frame SHALL contain the frame control field and an appropriate combination of routing fields as required.

In the frame control field, the frame type sub-field SHALL contain the value that indicates a NWK command frame, as shown in Table 3-48. All other sub-fields shall be set according to the intended use of the NWK command frame.

The routing fields SHALL contain an appropriate combination of address and broadcast fields, depending on the settings in the frame control field.

3.3.2.2.2 NWK Command Identifier Field

The NWK command identifier field indicates the NWK command being used. This field SHALL be set to one of the non-reserved values listed in Table 3-50.

3.3.2.2.3 NWK Command Payload Field

The NWK command payload field of a NWK command frame SHALL contain the NWK command itself.

3.4 Command Frames

The command frames defined by the NWK layer are listed in Table 3-50. The following sections detail how the NLME SHALL construct the individual commands for transmission.

For each of these commands, the following applies to the NWK header fields unless specifically noted in the section on NWK header in each command:

- The frame type sub-field of the NWK frame control field SHALL be set to indicate that this frame is a NWK command frame.
- The discover route sub-field in the NWK header SHALL be set to suppress route discovery (see Table 3-49).
- The source address field in the NWK header SHALL be set to the address of the originating device.

Table 3-50. NWK Command Frames

Command Frame Identifier	Command Name	Network Encryption	Reference
0x01	Route Request	Required	3.4.1
0x02	Route Reply	Required	3.4.2
0x03	Network Status	Required	3.4.3
0x04	Leave	Required	3.4.4
0x05	Route Record	Required	3.4.5
0x06	Rejoin Request	Optional	3.4.6
0x07	Rejoin Response	Optional	3.4.7
0x08	Link Status	Required	3.4.8
0x09	Network Report	Required	
0x0a	Network Update	Required	3.4.10
0x0b	End Device Timeout Request	Required	3.4.11
0x0c	End Device Timeout Response	Required	3.4.12
0x0d	Link Power Delta	Required	3.4.13
0x0e	Network Commissioning Request	Optional	3.4.14
0x0f	Network Commissioning Response	Optional	3.4.15
0x10 – 0xff	Reserved	-	—

3.4.1 Route Request Command

The route request command allows a device to request other devices within radio range to engage in a search for a particular destination device and establish a state within the network that will allow messages to be routed to that destination more easily and economically in the future. The payload of a route request command SHALL be formatted as illustrated in Figure 3-9.

Octets: 1	1	2	1	0/8	Variable
Command options	Route request identifier	Destination address	Path cost	Destination IEEE Address	TLVs
NWK command payload					

Figure 3-9. Route Request Command Frame Format

3.4.1.1 MAC Data Service Requirements

In order to transmit this command using the MAC data service, specified in IEEE Std 802.15.4-2020 [B1], the following information SHALL be included in the MAC frame header:

- The destination PAN identifier SHALL be set to the PAN identifier of the device sending the route request command.
- The destination address SHALL be set to the broadcast address of 0xffff.
- The source address and PAN identifier SHALL be set to the network address and PAN identifier of the device sending the route request command, which MAY or MAY NOT be the device from which the command originated.
- The frame control field SHALL be set to specify that the frame is a MAC data frame with MAC security disabled, since any secured frame originating from the NWK layer SHALL use NWK layer security. Because the frame is broadcast, no acknowledgment request SHALL be specified.
- The addressing mode and intra-PAN flags SHALL be set to support the addressing fields described here.

3.4.1.2 NWK Header Fields

In order for this route request to reach its destination and for the route discovery process to complete correctly, the following information SHALL be provided:

- The destination address in the NWK header SHALL be set to the broadcast address for all routers and the coordinator (see Table 3-76).
- The source IEEE address sub-field of the frame control field SHALL be set to 1 and the source IEEE address field of the NWK header SHALL be present and SHALL contain the 64-bit IEEE address of the originator of the frame.

3.4.1.3 NWK Payload Fields

The NWK frame payload contains a command identifier field, a command options field, the route request identifier field, the address of the intended destination, an up-to-date summation of the path cost, and the destination IEEE address.

The command frame identifier SHALL contain the value indicating a route request command frame.

3.4.1.3.1 Command Options Field

The format of the 8-bit command options field is shown in Figure 3-10.

Bit: 0-2	3-4	5	6	7
Reserved	Many-to-one	Destination IEEE address	Deprecated (Multicast)	Reserved

Figure 3-10. Route Request Command Options Field

3.4.1.3.1.1 Many-to-One

The many-to-one field SHALL have one of the non-reserved values shown in Table 3-51.

Table 3-51. Many-to-One Field Values

Value	Description
0	The route request is not a many-to-one route request.
1	The route request is a many-to-one route request and the sender supports a route record table.
2	The route request is a many-to-one route request and the sender does not support a route record table.
3	Reserved

3.4.1.3.1.2 Destination IEEE Address

The destination IEEE address field is a single-bit field. It SHALL have a value of 1 if, and only if, the command frame contains the destination IEEE address. The Destination IEEE Address field SHOULD always be added if it is known.

3.4.1.3.2 Route Request Identifier

The route request identifier is an 8-bit sequence number for route requests and is incremented by 1 every time the NWK layer on a particular device issues a route request.

3.4.1.3.3 Destination Address

The destination address SHALL be 2 octets in length and represents the intended destination of the route request command frame.

3.4.1.3.4 Path Cost

The path cost field is eight bits in length and is used to accumulate routing cost information as a route request command frame moves through the network (see section 3.6.4.5.2).

3.4.1.3.5 Destination IEEE Address

The destination IEEE address SHALL be 8 octets in length and represents the IEEE address of the destination of the route request command frame. It SHALL be present only if the destination IEEE address sub-field of the command frame options field has a value of 1.

3.4.2 Route Reply Command

The route reply command allows the specified destination device of a route request command to inform the originator of the route request that the request has been received. It also allows Zigbee routers along the path taken by the route request to establish state information that will enable frames sent from the source device to the destination device to travel more efficiently. The payload of the route reply command SHALL be formatted as illustrated in Figure 3-11.

Octets: 1	1	2	2	1	0/8	0/8	Variable
Command options	Route request identifier	Originator address	Responder address	Path cost	Originator IEEE address	Responder IEEE address	TLVs
NWK command payload							

Figure 3-11. Route Reply Command Format

3.4.2.1 MAC Data Service Requirements

In order to transmit this command using the MAC data service, specified in IEEE Std 802.15.4-2020 [B1], the following information SHALL be included in the MAC frame header:

The destination MAC address and PAN identifier SHALL be set to the network address and PAN identifier, respectively, of the first hop in the path back to the originator of the corresponding route request command frame. The destination PAN identifier SHALL be the same as the PAN identifier of the originator.

The source MAC address and PAN identifier SHALL be set to the network address and PAN identifier of the device sending the route reply command, which MAY or MAY NOT be the device from which the command originated.

The frame control field SHALL be set to specify that the frame is a MAC data frame with MAC security disabled, since any secured frame originating from the NWK layer SHALL use NWK layer security. The transmission options SHALL be set to require acknowledgment. The addressing mode and intra-PAN flags SHALL be set to support the addressing fields described here.

3.4.2.2 NWK Header Fields

In order for this route reply to reach its destination and for the route discovery process to complete correctly, the following information SHALL be provided:

- The source address in the NWK header SHALL be set to the 16-bit network address of the device transmitting the frame.
- The destination address field in the NWK header SHALL be set to the network address of the first hop in the path back to the originator of the corresponding route request.
- Since this is a NWK layer command frame, the source IEEE address sub-field of the frame control field SHALL be set to 1 and the source IEEE address field of the NWK header SHALL be present and SHALL contain the 64-bit IEEE address of the originator of the frame. The destination IEEE address sub-field of the frame control field SHALL also have a value of 1 and the destination IEEE address field of the NWK header shall be present and SHALL contain the 64-bit IEEE address of the first hop in the path back to the originator of the corresponding route request.
- The Sequence Number field in the NWK header SHALL be created for every hop during the route reply process. The Radius Field SHALL be set to *nwkcMaxDepth* * 2 by the target of the route request. Every hop during the Route Reply process SHALL decrement the radius by 1. If the value of the radius in the received Route Reply message is 1, the relaying router SHALL set the radius of the message to 1. The Sequence Number SHALL be created as if it were a new frame from the device transmitting the frame replacing the sequence number with

the device's next available sequence number. The Route Reply frame is not a forwarded frame, but is newly created by each hop during the route reply process.

3.4.2.3 NWK Payload Fields

The NWK frame payload contains a command identifier field, a command options field, the route request identifier, originator and responder addresses and an up-to-date summation of the path cost.

The command frame identifier SHALL contain the value indicating a route reply command frame.

3.4.2.3.1 Command Options Field

The format of the 8-bit command options field is shown in Figure 3-12.

Bit: 0 – 3	**4**	**5**	**6**	**6-7**
Reserved	Originator IEEE address	Responder IEEE address	Deprecated (Multicast)	Reserved

Figure 3-12. Route Reply Command Options Field

3.4.2.3.1.1 Originator IEEE Address

The originator IEEE address sub-field is a single-bit field. It SHALL have a value of 1 if and only if the originator IEEE address field is included in the payload. This bit SHALL always be set.

3.4.2.3.1.2 Responder IEEE Address

The responder IEEE address sub-field is a single-bit field. It SHALL have a value of 1 if, and only if, the responder IEEE address field is included in the payload. This bit SHALL always be set.

3.4.2.3.2 Route Request Identifier

The route request identifier is the 8-bit sequence number of the route request to which this frame is a reply.

3.4.2.3.3 Originator Address

The originator address field SHALL be 2 octets in length and SHALL contain the 16-bit network address of the originator of the route request command frame to which this frame is a reply.

3.4.2.3.4 Responder Address

The responder address field SHALL be 2 octets in length and SHALL always be the same as the value in the destination address field of the corresponding route request command frame.

3.4.2.3.5 Path Cost

The path cost field is used to sum link cost as the route reply command frame transits the network (see section 3.6.4.5.2).

3.4.2.3.6 Originator IEEE Address

The originator IEEE address field SHALL be 8 octets in length and SHALL contain the 64-bit address of the originator of the route request command frame to which this frame is a reply. This field SHALL only be present if the originator IEEE address sub-field of the command options field has a value of 1.

3.4.2.3.7 Responder IEEE Address

The responder IEEE address field SHALL be 8 octets in length and SHALL contain the 64-bit address of the destination of the route request command frame to which this frame is a reply. This field SHALL only be present if the responder IEEE address sub-field of the command options field has a value of 1.

3.4.3 Network Status Command

A device uses the network status command to report errors and other conditions arising in the NWK layer of a particular device to the peer NWK layer entities of other devices in the network. The NWK status command MAY be also used to diagnose network problems, *for example* address conflicts. The payload of a network status command SHALL be formatted as illustrated in Figure 3-13.

Octets: 1	**2**	**Variable**
Status code	Target address	TLVs
NWK command payload		

Figure 3-13. Network Status Command Frame Format

3.4.3.1 MAC Data Service Requirements

In order to transmit this command using the MAC data service, specified in IEEE Std 802.15.4-2020 [B1], the following information SHALL be provided:

- The destination MAC address and PAN identifier SHALL be set to the network address and PAN identifier, respectively, of the first hop in the path to the destination of the command frame or to the broadcast address 0xffff in the case where the command frame is being broadcast at the NWK layer.
- The source MAC address and PAN identifier SHALL be set to the network address and PAN identifier of the device sending the network status command.
- The frame control field SHALL be set to specify that the frame is a MAC data frame with MAC security disabled, since any secured frame originating from the NWK layer SHALL use NWK layer security. The transmission options SHALL NOT be set to require acknowledgement if the destination MAC address is the broadcast address 0xffff.
- The addressing mode and intra-PAN flags SHALL be set to support the addressing fields described here.

3.4.3.2 NWK Header Fields

Network status commands MAY be either unicast or broadcast. The fields of the NWK header SHALL be set as follows:

- The source address field SHALL always be set to the 16-bit network address of the device originating the command frame.
- The source IEEE address sub-field of the frame control field SHALL be set to 1 and the source IEEE address field of the NWK header SHALL be present and SHALL contain the 64-bit IEEE address of the originator of the frame.
- When sent in response to a routing error, the destination address field in the NWK header SHALL be set to the same value as the source address field of the data frame that encountered a forwarding failure.
- If and only if, the network status command frame is not broadcast, the destination IEEE address sub-field of the frame control field SHALL have a value of 1 and the destination IEEE address field of the NWK header SHALL be present and SHALL contain the 64-bit IEEE corresponding to the 16-bit network address in the destination address field if this IEEE address is known.

3.4.3.3 NWK Payload Fields

The NWK frame payload of the network status command frame contains a command frame identifier field, a status code field and a destination address field as described below. The command frame identifier SHALL be set to specify the network status command frame as defined in Table 3-52.

3.4.3.3.1 Status Code

The status code SHALL be set to one of the non-reserved values shown in Table 3-52.

Table 3-52. Status Codes for Network Status Command Frame

Value	Status Code	NLME-NWK-STATUS.indication Usage	Sent over-the-air in NWK Status Command	Description
0x00	Legacy No Route Available	No	Yes	This link code indicates a failure to route across a link. This was used in previous specifications. Revision 23 devices SHALL no longer SEND this error code but SHALL accept and act on it. It SHALL be treated the same as 0x02, Link failure.
0x01	Legacy Link Failure	No	Yes	This link code indicates a failure to route across a link. This was used in previous specifications. Revision 23 devices SHALL no longer SEND this error code but SHALL accept and act on it. It SHALL be treated the same as 0x02, Link failure.
0x02	Link failure	No	Yes	This link code indicates a failure to route across a link.
0x03 – 0x08	Deprecated	-	-	These are deprecated error codes and SHOULD NOT be used in a future specification version.
0x09	Parent link failure	Yes	No	The failure occurred as a result of a failure in the RF link to the

Value	Status Code	NLME-NWK-STATUS.indication Usage	Sent over-the-air in NWK Status Command	Description
				device's parent. This status is only used locally on a device to indicate loss of communication with the parent.
0x0A	Deprecated	-	-	These are deprecated error codes and SHOULD NOT be used in a future specification version.
0x0B	Source Route failure	Yes	Yes	Source routing has failed, probably indicating a link failure in one of the source route's links.
0x0C	Many-to-one route failure	Yes	Yes	A route established as a result of a many-to-one route request has failed.
0x0D	Address Conflict	Yes	Yes	The address in the destination address field has been determined to be in use by two or more devices.
0x0E	Deprecated	-	-	These are deprecated error codes and SHOULD NOT be used in a future specification version.
0x0F	PAN Identifier Update	Yes	No	The operational network PAN identifier of the device has been updated.
0x10	Network Address Update	Yes	No	The network address of the local device has been updated.
0x13	Unknown Command	No	Yes	The NWK command ID is not known to the device.

Value	Status Code	NLME-NWK-STATUS.indication Usage	Sent over-the-air in NWK Status Command	Description
0x14	PAN ID Conflict Report	Yes	No	Notification to the local application that a PAN ID Conflict Report has been received by the local Network Manager.
0x15 – 0xFF	Reserved	-	-	Reserved for future use

These status codes are used both as values for the status code field of a network status command frame and as values of the Status parameter of the NLME-NWK-STATUS.indication primitive.

A device receiving a reserved or deprecated status code SHALL ignore it.

3.4.3.3.2 Destination Address

The destination address field SHALL be 2 octets in length and SHALL be present if, and only if, the network status command frame is being sent in response to a routing failure or a network address conflict. In case of a routing failure, it SHALL contain the destination address from the data frame that encountered the failure; in case of an address conflict, it SHALL contain the offending network address.

3.4.4 Leave Command

The leave command is used by the NLME to inform other devices on the network that a device is leaving the network or else to request that a device leave the network. The payload of the leave command SHALL be formatted as shown in Figure 3-14.

1
Command options
NWK command payload

Figure 3-14. Leave Command Frame Format

3.4.4.1 MAC Data Service Requirement

In order to transmit this command using the MAC data service, specified in IEEE Std 802.15.4-2020 [B1], the following information SHALL be provided:

The destination MAC address and PAN identifier SHALL be set to the network address and PAN identifier, respectively, of the neighbor device to which the frame is being sent or else to the MAC broadcast address 0xffff in the case where the NWK header also contains a broadcast address.

The source MAC address and PAN identifier SHALL be set to the network address and PAN identifier of the device sending the leave command.

The frame control field SHALL be set to specify that the frame is a MAC data frame with MAC security disabled, since any secured frame originating from the NWK layer SHALL use NWK layer security. Acknowledgment SHALL be requested.

The addressing mode and intra-PAN flags SHALL be set to support the addressing fields described here.

3.4.4.2 NWK Header Fields

The NWK header fields of the leave command frame SHALL be set as follows:

- The source IEEE address sub-field of the frame control field SHALL be set to 1 and the source IEEE address field of the NWK header SHALL be present and SHALL contain the 64-bit IEEE address of the originator of the frame.
- If the request sub-field of the command options field is set to 1 then the destination address field in the NWK header SHALL be set to the network address of the child device being requested to leave.
- If the request sub-field is set to 0 then the destination address field in the NWK header SHALL be set to 0xfffd so that the indication is received by devices with *macRxOnWhenIdle* equal to TRUE.
- The destination address sub-field of the frame control MAY be set to 0 or 1. The choice SHALL be based on whether the local device has knowledge of the IEEE address for the device being requested to leave. If the local device knows the IEEE address then the field SHALL be set to 1 and the destination IEEE address field SHALL be present..
- The radius field SHALL be set to 1.

3.4.4.3 NWK Payload Fields

The NWK payload of the leave command frame contains a command frame identifier field and a command options field. The command frame identifier field SHALL be set to specify the leave command frame as described in Table 3-50.

3.4.4.3.1 Command Options Field

The format of the 8-bit Command Options field is shown in Figure 3-15.

Bit: 0 – 4	**5**	**6**	**7**
Reserved	Rejoin	Request	Remove children

Figure 3-15. Leave Command Options Field

3.4.4.3.1.1 Rejoin Sub-Field

The Rejoin sub-field is a single-bit field. If the value of this sub-field is 1, the device that is leaving from its current parent will rejoin the network. If the value of this sub-field is 0, the device will not rejoin the network.

3.4.4.3.1.2 Request Sub-Field

The request sub-field is a single-bit field. If the value of this sub-field is 1, then the leave command frame is a request for another device to leave the network. If the value of this sub-field is 0, then the leave command frame is an indication that the sending device plans to leave the network.

3.4.4.3.1.3 Remove Children Sub-Field

The remove children sub-field is a single-bit field. If this sub-field has a value of 1, then the children of the device that is leaving the network will also be removed. If this sub-field has a value of 0, then the children of the device leaving the network will not be removed.

3.4.5 Route Record Command

The route record command allows the route taken by a unicast packet through the network to be recorded in the command payload and delivered to the destination device. The payload of the route record command SHALL be formatted as illustrated in Figure 3-16.

Octets: 1	**Variable**
Relay count	Relay list
NWK command payload	

Figure 3-16. Route Record Command Format

3.4.5.1 MAC Data Service Requirements

In order to transmit this command using the MAC data service, specified in IEEE Std 802.15.4-2020 [B1], the following information SHALL be provided:

- The destination MAC address and PAN identifier SHALL be set to the network address and PAN identifier, respectively, of the neighbor device to which the frame is being sent.
- The source MAC address and PAN identifier SHALL be set to the network address and PAN identifier of the device sending the route record command.
- The frame control field SHALL be set to specify that the frame is a MAC data frame with MAC security disabled, since any secured frame originating from the NWK layer SHALL use NWK layer security. Acknowledgment SHALL be requested.
- The addressing mode and intra-PAN flags SHALL be set to support the addressing fields described here.

3.4.5.2 NWK Header Fields

The NWK header fields of the route record command frame SHALL be set as follows:

- If the route record is being initiated as the result of a NLDE-DATA.request primitive from the next higher layer, the source address field SHALL be set to the 16-bit network address of the originator of the frame. If the route record is being initiated as a result of the relaying of a data frame on behalf of one of the device's end device children, the source address field SHALL contain the 16-bit network address of that end device child.
- The source IEEE address sub-field of the frame control field SHALL be set to 1 and the source IEEE address field of the NWK header SHALL be present and SHALL contain the 64-bit IEEE address corresponding to the 16-bit network address contained in the source address field.
- The destination address field in the NWK header SHALL be set to the 16-bit network address of the concentrator device that is the destination of the frame.
- The destination IEEE address sub-field of the frame control field SHALL be set to 1, and the destination IEEE address field SHALL be set to the IEEE address of the concentrator device that is the destination of the frame, if this address is known.
- The Source Route sub-field of the frame control field SHALL be set to 0.

3.4.5.3 NWK Payload

The NWK frame payload contains a command identifier field, a relay count field, and a relay list field. The command frame identifier SHALL contain the value indicating a route record command frame.

3.4.5.3.1 Relay Count Field

This field contains the number of relays in the relay list field of the route record command. If the route record is being initiated as the result of a NLDE-DATA.request primitive from the next higher layer, the relay count field is initialized to 0. If the route record is being initiated as a result of the relaying of a data frame on behalf of one of the device's end device children, the relay count field is initialized to 1. In either case, it is incremented by each receiving relay.

3.4.5.3.2 Relay List Field

The relay list field is a list of the 16-bit network addresses of the nodes that have relayed the packet. If the route record is being initiated as a result of the relaying of a data frame on behalf of one of the device's end device children, the initiating device will initialize this field with its own 16-bit network address. Receiving relay nodes append their network address to the list before forwarding the packet.

3.4.6 Rejoin Request Command

The rejoin request command allows a device to rejoin its network. This is normally done in response to a communication failure, such as when an end device can no longer communicate with its original parent. The rejoin request command SHALL be formatted as shown in Figure 3-17.

Octets: 1
Capability Information
NWK command payload

Figure 3-17. Rejoin Request Command Frame Format

3.4.6.1 MAC Data Service Requirements

In order to transmit this command using the MAC data service, specified in IEEE Std 802.15.4.-2015, [B1], the following information SHALL be provided:

- The destination address and PAN identifier SHALL be set to the network address and PAN identifier, respectively, of the prospective parent.
- The source MAC address and PAN identifier SHALL be set to the network address and PAN identifier of the device transmitting the rejoin command frame.
- The transmission options SHALL be set to require acknowledgement.
- The addressing mode and intra-PAN flags SHALL be set to support the addressing fields described here.

3.4.6.2 NWK Header Fields

The NWK header fields of the rejoin request command frame SHALL be set as follows:

- The source address field of the NWK header to the 16-bit network address SHALL be as follows. If the value of the *nwkNetworkAddress* in the NIB is within the valid range, then it SHALL use that value. If the value of the *nwkNetworkAddress* in the NIB is not within the valid range, then it SHALL randomly generate a value within the valid range, excluding the value of 0x0000, and use that.
- The source IEEE address sub-field of the frame control field SHALL be set to 1, and the source IEEE address field SHALL be set to the IEEE address of the device issuing the request.
- The destination address field in the NWK header SHALL be set to the 16-bit network address of the prospective parent.

- The destination IEEE address sub-field of the frame control field SHALL be set to 1, and the destination IEEE address field SHALL be set to the IEEE address of the prospective parent, if this address is known.
- The radius field SHALL be set to 1.

3.4.6.3 NWK Payload Fields

The NWK frame payload contains a command identifier field and a capability information field. The command frame identifier SHALL contain the value indicating a rejoin request command frame.

3.4.6.3.1 Capability Information Field

This one-octet field has the format of the capability information field in the association request command in [B1], which is also described in Table 3-67.

3.4.7 Rejoin Response Command

The rejoin response command is sent by a device to inform a child of its network address and rejoin status. The rejoin request command SHALL be formatted as shown in Figure 3-18.

Octets: 2	**1**
Network address	Rejoin status
NWK command payload	

Figure 3-18. Rejoin Response Command Frame Format

3.4.7.1 MAC Data Service Requirements

In order to transmit this command using the MAC data service, specified in [B1], the following information SHALL be provided:

- The destination MAC address and PAN identifier SHALL be set to the network address and PAN identifier, respectively, of the device that sent the rejoin request to which this frame is a response.
- The source MAC address and PAN identifier SHALL be set to the network address and PAN identifier of the device that received and processed the rejoin request command frame.
- Acknowledgment SHALL be requested.
- The addressing mode and intra-PAN flags SHALL be set to support the addressing fields described here. The TXOptions SHALL request 'indirect transmission' to be used if the *Receiver on when idle* bit of the *nwkCapabilityInformation* contained in the corresponding rejoin request command is equal to 0x00. Otherwise, 'direct transmission' SHALL be used.

3.4.7.2 NWK Header Fields

The NWK header fields of the rejoin response command frame SHALL be set as follows:

- The source address field SHALL be set to the 16-bit network address of the device that is sending the response.
- The source IEEE address sub-field of the frame control field SHALL be set to 1 and the source IEEE address field of the NWK header SHALL be present and SHALL contain the 64-bit IEEE address of the parent device that is sending the response.
- The destination address field of the NWK header SHALL be set to the current network address of the rejoining device, *i.e.* the device that sent the join request to which this frame is a response.

- The destination IEEE address sub-field of the frame control field SHALL have a value of 1 and the destination IEEE address field of the NWK header SHALL be present and SHALL contain the 64-bit IEEE address of the child device that is source of the rejoin request command to which this frame is a response.
- The NWK layer will set the security of the Network Rejoin Response command frame to the same level as that of the received rejoin request command frame to which it is a response.

3.4.7.3 NWK Payload Fields

3.4.7.3.1 Network Address Field

If the rejoin was successful, this two-octet field contains the new network address assigned to the rejoining device. If the rejoin was not successful, this field contains the broadcast address (0xffff).

3.4.7.3.2 Rejoin Status Field

This field SHALL contain one of the non-reserved association status values specified in [B1].

3.4.8 Link Status Command

The link status command frame allows neighboring routers to communicate their incoming link costs to each other as described in section 3.6.4.4. Link status frames are transmitted as one-hop broadcasts without retries.

3.4.8.1 MAC Data Service Requirements

In order to transmit this command using the MAC data service, specified in IEEE Std 802.15.4-2020 [B1], the following information SHALL be included in the MAC frame header:

- The destination PAN identifier SHALL be set to the PAN identifier of the device sending the link status command.
- The destination address SHALL be set to the broadcast address of 0xffff.
- The source MAC address and PAN identifier SHALL be set to the network. address and PAN identifier of the device sending the link status command.
- The frame control field SHALL be set to specify that the frame is a MAC data frame with MAC security disabled, since any secured frame originating from the NWK layer SHALL use NWK layer security. Because the frame is broadcast, no acknowledgment request SHALL be specified.
- The addressing mode and intra-PAN flags SHALL be set to support the addressing fields described here.

3.4.8.2 NWK Header Fields

The NWK header field of the link status command frame SHALL be set as follows:

- The source IEEE address sub-field of the frame control field SHALL be set to 1 and the source IEEE address field of the NWK header SHALL be present and SHALL contain the 64-bit IEEE address of the originator of the frame.
- The destination address in the NWK header SHALL be set to the router-only broadcast address (see Table 3-76).
- The destination IEEE address sub-field of the frame control field SHALL have a value of 0 and the destination IEEE address field of the NWK header SHALL NOT be present.
- The radius field SHALL be set to 1.

3.4.8.3 NWK Payload Fields

The NWK command payload of the link status command SHALL be formatted as illustrated in Figure 3-19.

Octets: 1	Variable
Command options	Link status list
NWK command payload	

Figure 3-19. Link Status Command Format

3.4.8.3.1 Command Options Field

The format of the 8-bit command options field is shown in Figure 3-20.

Bit: 0 – 4	5	6	7
Entry count	First frame	Last frame	Reserved

Figure 3-20. Link Status Command Options Field

The entry count sub-field of the command options field indicates the number of link status entries present in the link status list. The first frame sub-field is set to 1 if this is the first frame of the sender's link status. The last frame sub-field is set to 1 if this is the last frame of the sender's link status. If the sender's link status fits into a single frame, the first frame and last frame bits SHALL both be set to 1.

3.4.8.3.2 Link Status List Field

An entry in the link status list is formatted as shown in Figure 3-21.

Octets: 2	1
Neighbor network address	Link status

Figure 3-21. Link Status Entry

Link status commands SHALL be transmitted on every active MAC interface in the MAC Interface table where the state is TRUE (active) and RoutersAllowed is also TRUE. The set of link status entries in the link status command derived from the neighbor table SHALL be specific to the interface that the command is to be transmitted on. Link status entries are sorted in ascending order by network address. If all router neighbors do not fit in a single frame, multiple frames are sent. When sending multiple frames, the last network address in the link status list for frame N is equal to the first network address in the link status list for frame N+1.

Each link status entry contains the network address of a router neighbor, least significant octet first, followed by the link status octet. The incoming cost field contains the device's estimate of the link cost for the neighbor, which is a value between 1 and 7. The outgoing cost field contains the value of the outgoing cost field from the neighbor table.

The link status field in a link status entry is formatted as in Figure 3-22.

Bits: 0-2	3	4-6	7
Incoming cost	Reserved	Outgoing cost	Reserved

Figure 3-22. Link Status Entry Format

3.4.9 Network Report Command

The network report command allows a device to report network events to the device identified by the address contained in the *nwkManagerAddr* in the NIB in an unsolicited way. Such events are radio channel condition and PAN ID conflicts. The payload of a network report command SHALL be formatted as illustrated in Figure 3-23.

Starting with Revision 23 of this specification this is considered a legacy command. Revision 23 devices SHALL NOT generate this command. Generating unsolicited messages on the network due to unencrypted traffic must be limited to avoid introducing security problems. Statistics on PAN ID conflicts are collected by the device and reported via the higher layer (such as ZDO).

Octets: 1	8	Variable
Command options (see Figure 3-24)	EPID	Report information
NWK command payload		

Figure 3-23. Network Report Command Frame Format

3.4.9.1 MAC Data Service Requirements

In order to transmit this command using the MAC data service, specified in [B1], the following information SHALL be included in the MAC frame header:

- The destination PAN identifier SHALL be set to the PAN identifier of the device sending the network report command.
- The destination address SHALL be set to the value of the next-hop address field in the routing table entry for which the destination address field has the same value as the *nwkManagerAddr* field in the NIB. If no such routing table entry exists, then the NWK MAY attempt route discovery as described in section 3.6.4.5.
- The source MAC address and PAN identifier SHALL be set to the network address and PAN identifier of the device sending the network report command, which MAY or MAY NOT be the device from which the command originated.
- The frame control field SHALL be set to specify that the frame is a MAC data frame with MAC security disabled, since any secured frame originating from the NWK layer SHALL use NWK layer security. The transmission options SHALL be set to require acknowledgment.

3.4.9.2 NWK Header Fields

The NWK header fields of the network report command frame SHALL be set as follows:

- The source IEEE address sub-field of the frame control field SHALL be set to 1 and the source IEEE address field of the NWK header SHALL be present and SHALL contain the 64-bit IEEE address of the originator of the frame.

- The destination address field in the NWK header SHALL be set to the 16-bit network address contained in the *nwkManagerAddr* attribute of the NIB.
- The destination IEEE address sub-field of the frame control field SHALL have a value of 1 and the destination IEEE address field of the NWK header SHALL be present and SHALL contain the 64-bit IEEE address of the corresponding to the 16-bit network address contained in the *nwkManagerAddr* attribute of the NIB, if this IEEE address is known.

3.4.9.3 NWK Payload Fields

The NWK frame payload contains a command identifier field, a command options field, an EPID field, and a report information payload.

The command frame identifier SHALL contain the value indicating a network report command frame.

3.4.9.3.1 Command Options Field

The format of the 8-bit command options field is shown in Figure 3-24.

Bits 0 - 4	5 - 7
Report information count	Report command identifier (see Figure 3-25)

Figure 3-24. Network Report Command Options Field

3.4.9.3.1.1 Report Information Count Sub-Field

The report information count sub-field contains an integer indicating the number of records contained within the Report Information field. The size of a record depends in the value of the Report Command Identifier.

3.4.9.3.1.2 Report Command Identifier Sub-Field

The report command identifier sub-field contains an integer indicating the type of report information command. Figure 3-25 contains the values that can be inserted into this field.

Report Command Identifier Value	Report Type
0x00	PAN identifier conflict
0x01 - 0x07	Reserved

Figure 3-25. Report Command Identifier Sub-Field

3.4.9.3.2 EPID Field

The EPID field SHALL contain the 64-bit EPID that identifies the network that the reporting device is a member of.

3.4.9.3.3 Report Information

The report information field provides the information being reported, the format of this field depends upon the value of the Report Command Identifier sub-field.

3.4.9.3.3.1 PAN Identifier Conflict Report

If the value of the Report Command Identifier sub-field indicates a PAN identifier conflict report then the Report Information field will have the format shown in Figure 3-26.

Octets: 2	2	2
1st PAN ID	...	nth PAN ID

Figure 3-26. PAN Identifier Conflict Report

The PAN ID conflict report SHALL be made up of a list of 16-bit PAN identifiers that are operating in the neighborhood of the reporting device. The number of PAN identifiers in the PAN ID conflict report SHALL be equal to the value of the report information count sub-field of the command options field.

3.4.10 Network Update Command

The network update command allows the device identified by the *nwkManagerAddr* attribute of the NIB to broadcast the change of configuration information to all devices in the network. For example, broadcasting the fact that the network is about to change its short PAN identifier.

The payload of a network update command SHALL be formatted as illustrated in Figure 3-27.

Octets: 1	8	1	Variable
Command Options (see Figure 3-28)	EPID	Update Id	Update Information
NWK command payload			

Figure 3-27. Network Update Command Frame Format

3.4.10.1 MAC Data Service Requirements

In order to transmit this command using the MAC data service specified in [B1], the following information SHALL be included in the MAC frame header:

- The destination PAN identifier SHALL be set to the old PAN identifier of the Zigbee coordinator in order for the command frame to reach network devices which have not received this update. The destination address SHALL be set according to the procedures for broadcast transmission outlined in section 3.6.6.
- The source MAC address and PAN identifier SHALL be set to the network address and the old PAN identifier of the device sending the network report command, which MAY or MAY NOT be the device from which the command originated.
- The frame control field SHALL be set to specify that the frame is a MAC data frame with MAC security disabled, since any secured frame originating from the NWK layer SHALL use NWK layer security.

3.4.10.2 NWK Header Fields

The NWK header fields of the network update command frame SHALL be set as follows:

- The source IEEE address sub-field of the frame control field SHALL be set to 1 and the source IEEE address field of the NWK header SHALL be present and SHALL contain the 64-bit IEEE address of the originator of the frame.
- The destination address in the NWK header SHALL be set to the broadcast address 0xffff.
- The destination IEEE address sub-field of the frame control field SHALL have a value of 0 and the destination IEEE address field SHALL NOT be present in the NWK header.

3.4.10.3 NWK Payload Fields

The NWK frame payload contains a command identifier field, a command options field, an EPID field and an Update Information variable field.

The command frame identifier SHALL contain the value indicating a network update command frame.

3.4.10.3.1 Command Options Field

The format of the 8-bit command options field is shown in Figure 3-28.

Bits 0 - 4	5 - 7
Update Information Count	Update Command Identifier (see Figure 3-29)

Figure 3-28. Network Update Command Options Field

3.4.10.3.1.1 Update Information Count Sub-Field

The update information count sub-field contains an integer indicating the number of records contained within the Update Information field. The size of a record depends on the value of the Update Command Identifier sub-field.

3.4.10.3.1.2 Update Command Identifier Sub-Field

The update command identifier sub-field contains an integer indicating the type of update information command. Figure 3-29 contains the values that can be inserted into this field.

Update Command Identifier Value	Report Type
0x00	PAN Identifier Update
0x01 – 0x07	Reserved

Figure 3-29. Update Command Identifier Sub-Field

3.4.10.3.2 EPID Field

The EPID field SHALL contain the 64bit EPID that identifies the network that is to be updated.

3.4.10.3.3 Update Id Field

The update Id field will reflect the current value of the *nwkUpdateId* attribute of the device sending the frame.

3.4.10.3.4 Update Information

The update information field provides the information being updated, the format of this field depends upon the value of the Update Command Identifier sub-field.

3.4.10.3.4.1 PAN Identifier Update

If the value of the Update Command Identifier sub-field indicates a PAN identifier update, then the Update Information field SHALL have the format shown in Figure 3-30.

Octets: 2
New PAN ID

Figure 3-30. PAN Identifier Update

The PAN identifier update SHALL be made up of a single 16-bit PAN identifier that is the new PAN identifier for this network to use. The Update Information count sub field SHALL be set equal to 1 as there is only a single PAN identifier contained within the Update Information field.

3.4.11 End Device Timeout Request Command

The End Device Timeout Request command is sent by an end device informing its parent of its timeout requirements. This allows the parent the ability to delete the child entry from the neighbor table if the child has not communicated with the parent in the specified amount of time.

The payload of an End Device Timeout Request command SHALL be formatted as illustrated in Figure 3-31.

Octets: 1	1
Request Timeout Enumeration	End Device Configuration

Figure 3-31. Format of the End Device Timeout Request Command

3.4.11.1 MAC Data Service Requirements

In order to transmit this command using the MAC data service, specified in [B1], the following information SHALL be provided:

- The destination address and PAN identifier SHALL be set to the network address and PAN identifier, respectively, of the end device's parent.
- The source MAC address and PAN identifier SHALL be set to the network address and PAN identifier of the device transmitting the End Device Timeout Request command.
- The transmission options SHALL be set to require acknowledgement.
- The address mode and intra-PAN flags SHALL be set to support the addressing fields described here.

3.4.11.2 NWK Header fields

The NWK header fields of the End Device Timeout Request command frame SHALL be set as follows:

- The source address field of the NWK header SHALL be set to the 16-bit network address.
- The source IEEE address sub-field of the frame control field SHALL be set to 1, and the source IEEE address field SHALL be set to the IEEE address of the device issuing the request.
- The destination address field in the NWK header SHALL be set to the 16-bit network address of the parent.
- The destination IEEE address sub-field of the frame control field SHALL be set to 1, and the destination IEEE address field SHALL be set to the IEEE address of the parent.
- The radius field SHALL be set to 1.

3.4.11.3 NWK Payload Fields

The NWK frame payload contains a command identifier field and the payload of the End Device Timeout Request as described in Table 3-53.

Table 3-53. Fields of the End Device Timeout Request

Name	Type	Valid Range	Description
Requested Timeout Enumeration	Enumerated type	0 – 14	The requested timeout enumeration. This will be converted into

			actual timeout value based on Table 2-54.
End Device Configuration	Bitmask	0x00 – 0x00	This is an enumeration of the child's requested configuration.

3.4.11.3.1 Requested Timeout Field

The valid values for the requested timeout will be an enumerated type between 0 and 14. This will be converted to an actual timeout value according to Table 3-54.

Table 3-54. Requested Timeout Enumerated Values

Requested Timeout Enumeration Value	Actual Timeout Value
0	10 seconds
1	2 minutes
2	4 minutes
3	8 minutes
4	16 minutes
5	32 minutes
6	64 minutes
7	128 minutes
8	256 minutes
9	512 minutes
10	1024 minutes
11	2048 minutes
12	4096 minutes
13	8192 minutes
14	16384 minutes

This allows for an actual timeout value between 10 seconds and 16384 minutes (~ 11 days).

3.4.11.3.2 End Device Configuration Field

Table 3-55. End Device Configuration Field Values

Bit	Description
0 – 15	Reserved for future use

This is a bitmask indicating the end device's requested configuration. At this time there are no enumerated bits in the configuration field. Devices adhering to this standard SHALL set the field to 0. To allow for future compatibility this field is left in place. Devices that receive the End Device Timeout Request message with an End Device Configuration field set to anything other than 0 SHALL reject the message.

This will allow parents to correctly report their lack of support for unknown end device features. The receiving device SHALL reject the request by sending an End Device Timeout Response with a status of 0x01 (UNSUPPORTED_FEATURE).

3.4.12 End Device Timeout Response Command

The End Device Timeout Response is sent by a router parent informing the end device whether it has accepted the timeout value that it was previously sent, and what its capabilities are.

Octets: 1	1
Status	Parent Information

Figure 3-32. Format of the End Device Timeout Response Command

3.4.12.1 MAC Data Service Requirements

In order to transmit this command using the MAC data service, specified in reference [B1], the following information SHALL be provided:

- The destination address and PAN identifier SHALL be set to the network address and PAN identifier, respectively, of the end device.
- The source MAC address and PAN identifier SHALL be set to the network address and PAN identifier of the device transmitting the End Device Timeout Response command.
- The transmission options SHALL be set to require acknowledgement.
- The address mode and intra-PAN flags SHALL be set to support the addressing fields described here.

3.4.12.2 NWK Header fields

The NWK header fields of the End Device Timeout Response command frame SHALL be set as follows:

- The source address field of the NWK header SHALL be set to the 16-bit network address.
- The source IEEE address sub-field of the frame control field SHALL be set to 1, and the source IEEE address field SHALL be set to the IEEE address of the device issuing the command.
- The destination address field in the NWK header SHALL be set to the 16-bit network address of the end device.
- The destination IEEE address sub-field of the frame control field SHALL be set to 1, and the destination IEEE address field SHALL be set to the IEEE address of the end device.
- The radius field SHALL be set to 1.

3.4.12.2.1 NWK Payload Fields

The NWK frame payload contains a command identifier field and a capability information field. The payload of the End Device Timeout Response is described in Table 3-56.

Table 3-56. Payload fields of the End Device Timeout Response

Name	Type	Valid Range	Description
Status	Enumeration	0 – 0xFF	The success or failure result of the previously received End Device Timeout Request command. See Table 3-57 for an enumeration of the status codes.
Parent Information	Bitmask	0 – 0xFF	This bitmask indicates the parent router's support information to the child device. The bitmask's values are described in Table 3-58.

Table 3-57. Enumeration of the End Device Timeout Response Status

Status	Value	Description
SUCCESS	0x00	The End Device Timeout Request message was accepted by the parent.
INCORRECT_VALUE	0x01	The received timeout value in the End Device Timeout Request command was outside the allowed range.
UNSUPPORTED_FEATURE	0x02	The requested feature is not supported by the parent router.
Reserved	0x03 – 0xFF	Reserved for future use.

Table 3-58. Values of the Parent Information Bitmask

Bits	Description
0	MAC Data Poll Keepalive Supported
1	End Device Timeout Request Keepalive Supported
2	Power Negotiation Support
3 – 15	Reserved for future use

3.4.13 Link Power Delta Command

The Link Power Delta command frame allows neighboring devices to communicate the value of the difference in dB between its optimal receive power level and the actual received power level (*ΔP)* of the last packet received with each other as described in section 3.4.13.7.

The Link power delta notification command frame also allows end devices to exchange the value of the difference in dB between its optimal receive power level and the actual received power level (*ΔP)* of the last packet received frame with its parent device as described in section 3.4.13.7.

3.4.13.1 MAC Data Service Requirements

Before any power negotiation has been performed, all transmissions SHALL be at the maximum transmit power. Once power levels have been negotiated as described in this section, all communications SHALL be at the last set power level. If the channel is changed or a rejoin performed, the joining SHALL be performed at the maximum power level.

The data transmission is done using the MAC data service, specified in [B1], the following information SHALL be included in the MAC frame header:

- The destination PAN identifier SHALL be set to the PAN identifier of the device sending the Link Power Delta command.
- The source MAC address and PAN identifier SHALL be set to the network address and PAN identifier of the device sending the Link Power Delta command.
- The destination address SHALL be set to the broadcast address of 0xffff when the Command Options field is set to Notification
- The destination address SHALL be set to the unicast destination address when the Command Options field is set to Request or Response.

- The frame control field SHALL be set to specify that the frame is a MAC data frame with MAC security disabled, since any secured frame originating from the NWK layer SHALL use NWK layer security.
- If the destination address of the frame is broadcast, no acknowledgment request SHALL be specified.
- If the destination address of the frame is a unicast network address, acknowledgment request SHALL be specified.
- The addressing mode and intra-PAN flags SHALL be set to support the addressing fields described here. The TxOptions SHALL request 'indirect transmission' to be used if the *Receiver on when idle* bit of the *nwkCapabilityInformation* contained in the NIB is 0x00. Otherwise, 'direct transmission' SHALL be used.

3.4.13.2 NWK Header Fields

The NWK header fields of the link power delta notification command frame SHALL be set as follows:

- The source address field of the NWK header SHALL be set to the 16-bit network address.
- The source IEEE address sub-field of the frame control field SHALL be set to 1, and the source IEEE address field SHALL be set to the IEEE address of the device issuing the request.
- If the sender is an end device, or responding to a request, the destination address field in the NWK header SHALL be set to the 16-bit network address of the parent. The destination IEEE address sub-field of the NWK frame control field SHALL be set to 1.
- If it is communicating power delta values for neighboring devices that have macRxOnWhenIdle = TRUE, the destination address in the NWK header SHALL be set to the macRxOnWhenIdle = TRUE broadcast address (see Table 3-64). In this case the destination IEEE address sub-field of the frame control field SHALL have a value of 0 and the destination IEEE address field of the NWK header SHALL NOT be present.
- The radius field SHALL be set to 1.

3.4.13.3 NWK Payload Fields

1 Octet	1 Octet	Variable
Command Options	List Count	Power List

Figure 3-33. NWK Payload Fields

3.4.13.4 Command Options Field

Bit: 0-1	2-7
Type	Reserved

Figure 3-34. Command Options Fields

Table 3-59. Command Options: Type Values

Value	Type	Description
0	Notification	An unsolicited notification. These frames are typically sent periodically from an RxOn device. If the device is a FFD, it is broadcast to all RxOn devices (0xfffd), and includes power information for all neighboring RxOn devices. If the device is an RFD with RxOn, it is sent unicast to its Parent, and includes only power information for the Parent device.
1	Request	Typically used by sleepy RFD devices that do not receive the periodic Notifications from their Parent. The sleepy RFD will wake up periodically to send this frame to its Parent, including only the Parent's power information in its payload. Upon receipt, the Parent sends a Response (Type = 2) as an indirect transmission, with only the RFD's power information in its payload. After macResponseWaitTime, the RFD polls its Parent for the Response, before going back to sleep. Request commands are sent as unicast. Note: any device MAY send a Request to solicit a Response from another device. These commands SHALL be sent as unicast and contain only the power information for the destination device. If this command is received as a broadcast, it SHALL be discarded with no action.
2	Response	This command is sent in response to a Request. Response commands are sent as unicast to the sender of the Request. The response includes only the power information for the requesting device.
3	Reserved	

3.4.13.5 List Count

Number of power delta records in the power list.

3.4.13.6 Power List

2 Octets	1 Octet
Device Address	Power Delta

Figure 3-35. Power List

3.4.13.6.1 Device Address

Network address of the device whose power delta is conveyed in this notification.

3.4.13.6.2 Delta Power

Delta power (ΔP) calculated as *Popt – Prx*. This is the value of the difference in dB between its optimal receive power level (*Popt*) and the actual received power level (*Prx*) of the last packet received.

3.4.13.7 Link Power Delta command behavior

When joined to a network, a Zigbee router or coordinator that supports Power Control SHALL periodically send a Link Power Delta command with Type = Notification (0), every nwkLinkPowerDeltaTransmitRate seconds plus a one off random jitter of between 0 and 10 seconds, as a one-hop broadcast (0xfffd) without retries. A value of 0 for nwkLinkPowerDeltaTransmitRate indicates that Link Power Delta commands are never sent. It is allowed for End Devices to use a value other than the default rate to reduce the transmission rate and save battery life.

An end device that supports Power Control SHALL generate a Link Power Delta message only if the nwkParentInformation in the NIB indicates bit 2 is set to 1, meaning the parent supports Power Negotiation. The Link Power Delta SHALL be sent as follows:

1. The message SHALL be unicast to the router parent of the end device.
2. The message SHALL only contain the router parent information in the Link Power Delta message.

The Power List SHALL contain all active devices in its neighbor table with macRxOnWhenIdle = TRUE. Multiple Link Power Delta commands MAY be sent if not all the devices from the neighbor table can fit within a single frame. Subsequent commands SHOULD have additional random jitter applied.

When joined to a network, a Zigbee end device with macRxOnWhenIdle = TRUE and that supports Power Control , SHALL periodically send a Link Power Delta command with Type = Notification (0) as a unicast its Parent, every *nwkLinkPowerDeltaTransmitRate* seconds plus a one off random jitter of between 0 and 10 seconds. The Power List SHALL contain only the Parent.

When joined to a network, a Zigbee end device with macRxOnWhenIdle = FALSE and that supports Power Control, SHALL periodically wake up and send a Link Power Delta command with Type = Request (1) as a unicast to its Parent, every *nwkLinkPowerDeltaTransmitRate* seconds plus a one off random jitter of between 0 and 10 seconds. The Power List SHALL contain only the Parent device. The end device SHALL wait *macResponseWaitTime* before polling its Parent for the link power delta command with Type = Response (2). The Power List in the Response SHALL contain only the end device.

The Power Delta to be included for each device in the Power List SHALL be the difference in dBm between the optimal level (defined as 20 dB above the sensitivity requirement, see Annex D.9.2.4.2) and the last available RSSI for that device.

Upon receipt of a Link Power Delta command, a device that supports Power Control SHALL do the following.

1. Find an entry in the *nwkNeighborTable* where the NWK Source Address of the Link Power Delta command corresponds to the Network Address value of the entry. If no entry is found, the message SHALL be dropped and no further processing SHALL be done.
2. Examine Link Power Delta command and find the Device Address in the payload of the message that matches the *nwkNetworkAddress* value in its NIB. If no match is found and the receiving device is an End Device, then the message SHALL be dropped and no further processing SHALL be done.

3. Using the MLME of the MAC interface that the message arrived on, execute a MLME-SET-POWER-INFORMATION-TABLE.request with the following parameters.
    a. Set the Short address to the NWK Source of the Link Power Delta Command.
    b. Set the IEEE address to the Source IEEE of the Link Power Delta Command.
    c. Set the TX Power level as described from section D.11.2.
    d. Set the last RSSI level according to the RSSI parameter of the MCPS-DATA.indication.
4. If the receiving device is an end device, processing is complete. No further processing SHALL be done.
5. If the receiving device is a router, it SHALL do the following.
    a. If the entry in the *nwkNeighborTable* indicates a Device Type value other than 0x02 (Zigbee End Device), processing is complete. No further processing SHALL be done.
    b. Otherwise this message is from an End Device child of the router. The router SHALL generate a response Link Power Delta Command accordingly:
    c. The NWK destination SHALL be the NWK Source of the received Link Power Delta Command, not a broadcast address.

3.4.14 Network Commissioning Request Command

The Network Commissioning Request command allows a device to request joining or rejoining to the network. This MAY be used for negotiating a dynamic link key prior to joining or rejoining, or it can be used to join or rejoin and receive a transport key sent by the trust center using the device's existing link key [PICS-NWK-ASSOCIATE-REQUEST.1].

This command SHALL be the preferred mechanism to join or rejoin when both sender and receiver support it.

If the nwkNetworkAddress value of the NIB is unset, the device SHALL generate a random short address. That value SHALL be used for sending this command frame.

The Network Commissioning Request Command SHALL be formatted as shown in Figure 3-36.

Octets: 1	1	Variable
Network Commissioning Type	Capability Information	Zigbee TLVs
Network Command Payload		

Figure 3-36. Network Commissioning Request Command Format

3.4.14.1 MAC Data Service Requirements

In order to transmit this command using the MAC data service, specified in IEEE-Std 802.15.4-2020, [B1], the following information SHALL be provided [PICS-NWK-ASSOCIATE-REQUEST.2]:

- The destination address and PAN identifier SHALL be set to the network address and PAN identifier, respectively, of the prospective parent.
- The source MAC address and PAN identifier SHALL be set to the network address and PAN identifier of the device transmitting the rejoin command frame.
- The destination and source address modes SHALL be set to short.
- The transmission options SHALL be set to require acknowledgement.
- The address mode and intra-PAN flags SHALL be set to support the addressing fields described above.

3.4.14.2 NWK Header Fields

The NWK header fields of the rejoin request command frame SHALL be set as follows:

The source address field of the NWK header to the 16-bit network address SHALL be as follows. If the value of the *nwkNetworkAddress* in the NIB is within the valid range, then it SHALL use that value. If the value of the *nwkNetworkAddress* in the NIB is not within the valid range, then it SHALL randomly generate a value within the valid range, excluding the value of 0x0000, and use that.

- The source IEEE address sub-field of the frame control field SHALL be set to 1, and the source IEEE address field SHALL be set to the IEEE address of the device issuing the request.
- The destination address field in the NWK header SHALL be set to the 16-bit network address of the prospective parent.
- The destination IEEE address sub-field of the frame control field SHALL be set to 1, and the destination IEEE address field SHALL be set to the IEEE address of the prospective parent, if this address is known.
    - The radius field SHALL be set to 1.

3.4.14.3 NWK Payload Fields

The NWK frame payload contains a command identifier field, a capability information field, and one or more TLVs. The command frame identifier SHALL contain the value indicating a network associate command frame.

3.4.14.3.1 Network Commissioning Type

Table 3-60 defines the Commissioning Types that can be used.

Table 3-60. Network Commissioning Types

ID	Description
0x00	Initial Join
0x01	Rejoin

3.4.14.3.2 Capability Information Field

This one-octet field has the format of the capability information field in the association request command in [B1], which is also described in Table 3-67.

3.4.14.3.3 TLVs

The remainder of this message MAY contain one or more TLVs as defined by Zigbee. The total size of the TLVs SHALL NOT exceed *capsJoinerTLVsUnfragmentedMaxSize* bytes. This allows for the APS Update Device message sent by the parent router to fit the TLV data without fragmentation.

The device sending the Network Commissioning Request command communicates information to the parent device by including TLVs directly in the message. The device SHALL include the Joiner Encapsulation Global TLV. The remainder of this message MAY contain other TLVs as defined by Zigbee. In a multi-hop joining scenario the Trust Center and parent device will not be the same entity. Information about the sending device is communicated to the Trust Center through the Joiner Encapsulation Global TLV, which will be relayed in its entirety. To avoid fragmentation when forwarding TLV data to the Trust Center via APS UpdateDevice message from a parent router, the total size of TLVs SHALL NOT exceed apscJoinerTlvsUnfragmentedMaxSize bytes.

When a device creates the Joiner Encapsulation Global TLV it SHALL contain the following TLVs inside it:

- Fragmentation Parameters Global TLV
- If the device is not rejoining: Supported Key Negotiation Methods Global TLV

At this time this Revision of the specification does not support negotiating a new link key during rejoin. Therefore, devices certified to this Revision SHALL not include the Supported Key Negotiation Methods Global TLV inside the Joiner Encapsulation TLV so it is clear to the Trust Center that the device does not support this behavior. Future revisions of this specification that support this would include this TLV as a clear sign the rejoining device supports this new functionality.

Additional TLVs MAY be included inside the Joiner Encapsulation Global TLV to be relayed to the Trust Center or MAY be included outside the Joiner Encapsulation Global TLV to be communicated only to the parent router.

The General TLV Processing rules in section I.4.8 SHALL be executed on receipt of the Network Commissioning Request Command frame.

3.4.15 Network Commissioning Response Command

The Network Commissioning Response command is sent by a device to inform a requesting device of its network address and network commissioning request status. The Network Commissioning Response command SHALL be formatted as shown in Figure 3-37.

Octets: 2	1
Network address	Status
NWK Command payload	

Figure 3-37. Network Commissioning Response Format

3.4.15.1 MAC Data Service Requirements

In order to transmit this command using the MAC data service, specified in [B1], the following information SHALL be provided:

- The destination MAC address and PAN identifier SHALL be set to the network address and PAN identifier, respectively, of the device that sent the Network Commissioning Response to which this frame is a response.
- The source MAC address and PAN identifier SHALL be set to the network address and PAN identifier of the device that received and processed the Network Commissioning Response command frame.
- Acknowledgment SHALL be requested.
- The addressing mode and intra-PAN flags SHALL be set to support the addressing fields described here. The TXOptions SHALL request ‘indirect transmission’ to be used if the Receiver on when idle bit of the *nwkCapabilityInformation* contained in the corresponding Network Commissioning Request command is equal to 0x00. Otherwise, ‘direct transmission’ SHALL be used.

3.4.15.2 NWK Header Fields

The NWK header fields of the rejoin response command frame SHALL be set as follows:

- The source address field SHALL be set to the 16-bit network address of the device that is sending the response.
- The source IEEE address sub-field of the frame control field SHALL be set to 1 and the source IEEE ad-dress field of the NWK header SHALL be present and SHALL contain the 64-bit IEEE address of the parent device that is sending the response.
- The destination address field of the NWK header SHALL be set to the current network address of the device that sent the NWK Commissioning Request frame, i.e. the device that sent the join request to which this frame is a response.
- The destination IEEE address sub-field of the frame control field SHALL have a value of 1 and the destination IEEE address field of the NWK header SHALL be present and SHALL contain the 64-bit IEEE address of the child device that is source of the Network Commissioning Request frame.
- The NWK layer will set the security of the Network Commissioning Response frame to the same level as that of the received Network Commissioning Request frame.

3.4.15.2.1 NWK Payload Fields

3.4.15.2.1.1 Network Address Field

If the network commissioning request was successful, this two-octet field contains the network address assigned to the device and will be the same as the value used in the Network Commissioning Request. This address could be different than the value used for the Network & MAC Destination header fields if the requesting device's address is already being used on the network. In that case, the Status field will also contain value of 0xF0, indicating that the commissioning request has not succeeded due to address conflict, but the device should retry the operation with the new address. If the network commissioning was not successful and should not be retried, this field contains the broadcast address (0xffff).

3.4.15.2.1.2 Status Field

In the special case of an address conflict the status SHALL be the value 0xF0, which is normally a reserved value for the association status in [B1]. In this context it indicates a short address conflict. The receiving device can retry the operation using the new short address specified in the Network Address field. Otherwise, this field SHALL contain one of the non-reserved association status values specified in [B1]. Refer to section 3.6.1.6.1.3 for further clarification on selecting a status value.

3.5 Constants and NIB Attributes

3.5.1 NWK Constants

The constants that define the characteristics of the NWK layer are presented in Table 3-61.

Table 3-61. NWK Layer Constants

Constant	Description	Value
nwkcCoordinatorCapable	A Boolean flag indicating whether the device is capable of becoming the Zigbee coordinator. A value of 0x00 indicates that the device is not capable of becoming a coordinator while a value of 0x01 indicates that the device is capable of becoming a coordinator.	Configuration dependent
nwkcMinHeaderOverhead	The minimum number of octets added by the NWK layer to an NSDU.	0x08
nwkcProtocolVersion	The version of the Zigbee NWK protocol in the device.	0x02
nwkcRouteDiscoveryTime	The number of OctetDurations until a route discovery expires.	0x4c4b4 (0x2710 msec on 2.4GHz)
nwkcMaxBroadcastJitter	The maximum broadcast jitter time measured in OctetDurations.	0x7d0 (0x40 msec on 2.4GHz)

Constant	Description	Value
nwkcInitialRREQRetries	The number of times the first broadcast transmission of a route request command frame is retried.	0x03
nwkcRREQRetries	The number of times the broadcast transmission of a route request command frame is retried on relay by an intermediate Zigbee router or Zigbee coordinator.	0x02
nwkcRREQRetryInterval	The number of OctetDurations between retries of a broadcast route request command frame.	0x1f02 (0xfe msec on 2.4Ghz)
nwkcMinRREQJitter	The minimum jitter, in OctetDurations, for broadcast retransmission of a route request command frame.	0x3f (2 msec on 2.4GHz)
nwkcMaxRREQJitter	The maximum jitter, in OctetDurations, for broadcast retransmission of a route request command frame.	0xfa0 (128 msec on 2.4GHz)
nwkcMACFrameOverhead	The size of the MAC header used by the Zigbee NWK layer.	0x0b
nwkcMaxDepth	The maximum depth of the network (number of hops) used for various calculations of network timing and limitations.	15
nwkcUnicastRetries	The number of network layer retries on unicast messages that are attempted before reporting the result to the higher layer.	3
nwkcUnicastRetryDelay	The delay between network layer retries.	50 ms
nwkcMinRouterBootstrapJitter	The minimum jitter, in OctetDurations, for transmission of a gratuitous link status message. Refer to section 3.6.4.4.2 for further clarification on link status message transmission.	0x3d09 (500 msec on 2.4GHz)
nwkcMaxRouterBootstrapJitter	The maximum jitter, in OctetDurations, for transmission of a gratuitous link status message.	0x7a12 (1 sec on 2.4GHz)
nwkcBroadcastDeliveryTime	The total delivery time for a broadcast transmission to be delivered to all RxOnWhenIdle=TRUE devices in the network.	9 seconds

3.5.2 NWK Information Base

The NWK information base (NIB) comprises the attributes required to manage the NWK layer of a device. Each of these attributes can be read or written using the NLME-GET.request and NLME-SET.request primitives, respectively, except for attributes for which the Read Only column contains a value of Yes. In that case, the attributes value MAY be read using the NLME-GET.request primitive but MAY NOT be set using the NLME-SET.request primitive. Generally, these read-only attribute are set using some other mechanism. For example, the *nwkSequenceNumber* attribute is set as specified in section 3.6.2.1 and incremented every time the NWK layer sends a frame. The attributes of the NIB are presented in Table 3-62.

Table 3-62. NIB Attributes

Attribute	Id	Type	Read Only	Range	Description	Default
nwkSequenceNumber	0x81	Integer	Yes	0x00 – 0xff	A sequence number used to identify outgoing frames (see section 3.6.2).	Random value from within the range
nwkPassiveAckTimeout	0x82	Integer	No	0x000000 – 0xffffff	The maximum time duration in OctetDurations allowed for the parent and all child devices to retransmit a broadcast message (passive acknowledgment timeout).	500 ms
nwkMaxBroadcastRetries	0x83	Integer	No	0x00 – 0x5	The maximum number of retries allowed after a broadcast transmission failure.	0x02
nwkMaxChildren	0x84	Integer	No	0x00 – 0xff	The number of children a device is allowed to have on its current network. Note that the value of this attribute is implementation-dependent.	Implementation-dependent
nwkcMaxDepth	0x85	Integer	Yes	0x00 – 0xff	The depth a device can have.	15
Deprecated	0x86					
nwkNeighborTable	0x87	Set	No	Variable	The current set of neighbor table entries in the device (see Table 3-71).	Null set

Attribute	Id	Type	Read Only	Range	Description	Default
nwkNetworkBroadcastDeliveryTime	0x88	Integer	No	0 – 0xffffffff	Time duration in OctetDurations that a broadcast message needs to encompass the entire network. This is a calculated quantity based on other NIB attributes.	Defined in stack profile
Deprecated	0x89					
Deprecated	0x8a					
nwkRouteTable	0x8b	Set	No	Variable	The current set of routing table entries in the device (see Table 3-73).	Null set
Deprecated	0x8e					
nwkCapabilityInformation	0x8f	Bit vector	Yes	See Table 3-67.	This field SHALL contain the device capability information established at network joining time.	0x00
Deprecated	0x90					
Deprecated	0x91					
nwkManagerAddr	0x92	Integer	No	0x0000 – 0xfff7	The address of the designated network channel manager function.	0x0000
nwkMaxSourceRoute	0x93	Integer	No	0x00 – 0xff	The maximum number of hops in a source route.	0x0c
nwkUpdateId	0x94	Integer	No	0x00 – 0xff	The value identifying a snapshot of the network settings with which this node is operating with.	0x00
nwkcTransactionPersistenceTime	0x95	Integer	No	0x0000 – 0xffff	The maximum time (in superframe periods) that a transaction is stored by a coordinator and indicated in its beacon. This attribute reflects the value of the MAC PIB attribute	7680 ms

Attribute	Id	Type	Read Only	Range	Description	Default
					macTransactionPersistenceTime (see [B1]) and any changes made by the higher layer will be reflected in the MAC PIB attribute value as well.	
nwkNetworkAddress	0x96	Integer	No	0x0000 – 0xfff7	The 16-bit address that the device uses to communicate with the PAN. This attribute reflects the value of the MAC PIB attribute *macShortAddress* (see [B1]) and any changes made by the higher layer will be reflected in the MAC PIB attribute value as well.	0xffff
nwkStackProfile	0x97	Integer	No	0x00 – 0x0f	The identifier of the Zigbee stack profile in use for this device.	
nwkBroadcastTransactionTable	0x98	Set	Yes	-	The current set of broadcast transaction table entries in the device (see Table 3-77).	Null set
Deprecated	0x99					
nwkExtendedPANID	0x9a	64-bit extended address	No	0x0000000000000000 – 0xfffffffffffffffe	The Extended PAN Identifier for the PAN of which the device is a member. The value 0x0000000000000000 means the Extended PAN Identifier is unknown.	0x0000000000000000
Deprecated	0x9b					
nwkRouteRecordTable	0x9c	Set	No	Variable	The route record table (see Table 3-63).	Null Set

Attribute	Id	Type	Read Only	Range	Description	Default
nwkIsConcentrator	0x9d	Boolean	No	TRUE or FALSE	A flag determining if this device is a concentrator. This only applies when the device is operating as a Concentrator. TRUE = Device is a concentrator. FALSE = Device is not a concentrator.	FALSE
nwkConcentratorRadius	0x9e	Integer	No	0x00 – 0xff	The hop count radius for concentrator route discoveries. This only applies when the device is operating as a Concentrator. This only applies when the device is operating as a Concentrator.	0x0000
nwkConcentratorDiscoveryTime	0x9f	Integer	No	0x00 – 0xff	The time in seconds between concentrator route discoveries. If set to 0x0000, the discoveries are done at start up and by the next higher layer only. This only applies when the device is operating as a Concentrator.	0x0000
nwkSecurityLevel	0xa0		No		Security attribute defined in Chapter 4.	
nwkSecurityMaterialSet	0xa1		No		Security attribute defined in Chapter 4.	
nwkActiveKeySeqNumber	0xa2		No		Security attribute defined in Chapter 4.	
nwkAllFresh	0xa3		No		Security attribute defined in Chapter 4.	
nwkConcentratorDiscoverySeparationTime	0xa4	Integer	No	0x00 – 0xff	The minimum time, in seconds, between two consecutive concentrator route discoveries. If set to 0x00, there is no minimum separation. This only applies when the	

Attribute	Id	Type	Read Only	Range	Description	Default
					device is operating as a Concentrator.	
nwkLinkStatusPeriod	0xa6	Integer	No	0x00 – 0xff	The time in seconds between link status command frames.	0x0f
nwkRouterAgeLimit	0xa7	Integer	No	0x00 – 0xff	The number of missed link status command frames before resetting the link costs to zero.	3
Deprecated	0xa8					
nwkAddressMap	0xa9	Set	No	Variable	The current set of 64-bit IEEE to 16-bit network address map (see Table 3-64).	Null Set
nwkTimeStamp	0x8C	Boolean	No	TRUE or FALSE	A flag that determines if a time stamp indication is provided on incoming and outgoing packets. TRUE= time indication provided. FALSE = no time indication provided.	FALSE
nwkPANId	0x80	16-bit PAN ID	No	0x0000 – 0xffff	This NIB attribute should, at all times, have the same value as *macPANId*.	0xffff
nwkTxTotal	0x8D	Integer	No	0x0000 – 0xffff	A count of unicast transmissions made by the NWK layer on this device. Each time the NWK layer transmits a unicast frame, by invoking the MCPS-DATA.request primitive of the MAC sub-layer, it SHALL increment this counter. When either the NHL performs an NLME-SET.request on this attribute or if the value of nwkTxTotal rolls over past 0xffff the	0

Attribute	Id	Type	Read Only	Range	Description	Default
					NWK layer SHALL reset to 0x00 each Transmit Failure field contained in the neighbor table.	
nwkLeaveRequestAllowed	0xAA	Boolean	No	TRUE or FALSE	This policy determines whether or not a remote NWK leave request command frame received by the local device is accepted.	TRUE
nwkParentInformation	0xAB	Bitmask	No	0x00 – 0xFF	The behavior depends upon whether the device is an FFD or RFD. For an RFD, this records the information received in an End Device Timeout Response command indicating the parent information. The bitmask values are defined in Table 3-58. For an FFD, this records the device's local capabilities.	0x00
nwkEndDeviceTimeoutDefault	0xAC	Integer	No	0x00 – 0xFF	This is an index into Table 3-54. It indicates the default timeout in minutes for any end device that does not negotiate a different timeout value.	8

Attribute	Id	Type	Read Only	Range	Description	Default
nwkLeaveRequestWithoutRejoinAllowed	0xAD	Boolean	No	TRUE or FALSE	This policy determines whether a NWK leave request is accepted when the Rejoin bit in the message is set to FALSE	TRUE
nwkIeeeAddress	0xAE	64-bit address	Yes	0x0000000000000001 – 0xFFFFFFFFFFFFFFFF	The IEEE address of the local device.	
nwkMacInterfaceTable	0xAF	Set	No	Variable	A table of lower-layer interfaces managed by the network layer. See Table 3-65.	
nwkNetworkWideBeaconAppendixTLVs	0xB0	Array	No	0 – 127 bytes	This is a list of TLVs that are global to the Zigbee network.	0 byte length array
nwkDeviceLocalBeaconAppendixTLVs	0xB1	Array	No	0-127 bytes	This is a list of TLVs that are specific to the local device. It is mandatory for routers to always include the Router Information TLV.	Router Information TLV
Deprecated	0xB2					
Deprecated	0xB3					
nwkDiscoveryTable	0xB4	Array	No	Varies	This stores the set of potential networks and parents that the device is considering when joining or rejoining. See Table 3-64 for the fields of each entry.	None
nwkDiscoveryTableSize	0xB5	Integer	Yes	6 – 100	The number of entries the nwkDiscoveryTable can hold.	6

Attribute	Id	Type	Read Only	Range	Description	Default
nwkNextPanId	0xB6	Integer	No	0x0000 – 0xFFFF	This indicates what the next PAN ID received in the NWK Update Command frame SHALL be in order for a PAN ID change to be accepted. A value of 0xFFFF allows any PAN ID to be accepted.	0xFFFF
nwkNextChannelChange	0xB7	ChannelPage Structure	No	Any valid	This indicates the next channel that will be used once a command to change channels has been received. A value of 0 indicates any channel is valid as the next channel.	0
nwkPerformAdditionalMacDataPollRetries	0xB9	Integer	No	0 – 10	This indicates that the network layer will perform additional attempts upon receipt of a MAC Data poll failure.	0
Reserved	0xBA					
Reserved	0xBB					
nwkPreferredParent	0xBC	Boolean	No	TRUE or FALSE	Indicates to a potential child capacity to act as a parent as defined by a next higher-level application. Defaults to FALSE for routers that do not make a determination.	
nwkHubConnectivity	0xBD	Boolean	No	TRUE or FALSE	This indicates whether the router has Hub Connectivity as defined by a higher level application. The higher level application sets this value and the stack advertises it.	FALSE

Attribute	Id	Type	Read Only	Range	Description	Default
nwkRoutingSequenceNumber	0xBE	Integer	Yes	0-0xFFFF	A strictly increasing sequence number included in all route request and route reply command frames to allow other routers to determine the chronological order of such route discovery messages.	Previously persisted value; 0 when factory-new
nwkGoodParentLQA	0xBF	Integer	Yes	0 – 255	This indicates the lowest LQA value for beacons received from routers so that they will be preferred for joining or rejoining. LQI is used instead of LQA when Active Power Control is used on this link. See section 3.6.1.5.2 for its usage.	75
nwkPanIdConflictCount	0xC0	Integer	Yes	0 – 65,535	This indicates the total number of PAN ID conflicts that have been seen by the local device. This value can be reset to 0 by the higher layer. This value SHALL be required for routers and coordinators. It is optional for End Devices.	0
nwkMaxInitialJoinParentAttempts	0xC2	Integer	No	0 – 255	The maximum number of attempts to join parent devices for a particular network.	1
nwkMaxRejoinParentAttempts	0xC3	Integer	No	0 – 255	The maximum number of attempts to rejoin to parent devices for the current network.	3

7647

Table 3-63. Route Record Table Entry Format

Field Name	Field Type	Valid Range	Reference
Network Address	Integer	0x0000 – 0xfff7	The destination network address for this route record.
Relay Count	Integer	0x0000 – 0xffff	The count of relay nodes from concentrator to the destination.
Path	Set of Network Addresses		The set of network addresses that represent the route in order from the concentrator to the destination.

Table 3-64. Network Address Map

64-bit IEEE Address	16-bit Network Address
A valid 64-bit IEEE Address or Null if not known	0x0000 – 0xfff7

Table 3-65. Fields of the MAC Interface Table (nwkMacInterfaceTable)

Field Name	Field Type	Valid Range	Description
Index	Integer	0 – 31	A unique index that can be used to identify an entry in this table.
State	Boolean	TRUE or FALSE	A Boolean indicating whether the interface is currently enabled for sending and receiving messages. TRUE indicates the interface is enabled, FALSE means it is disabled.
Supported Channels	Channel List Structure	Varies	A Channel List Structure indicating the pages and channels that are supported by this interface. **NOTE: The interfaces SHALL have mutually exclusive Supported Channels lists.**

Field Name	Field Type	Valid Range	Description
Channel In Use	Channel Page Structure	Varies	The current channel in use by the device. Only a single channel in the Channel Page Structure MAY be selected at one time.
RoutersAllowed	Boolean	TRUE or FALSE	A Boolean indicating whether routers are allowed to join to this device on this interface.
nwkLinkPowerDeltaTransmitRate	Integer	0 – 65,535	The rate, in seconds, of how often a Link Power Delta request is generated. In bands where this is optional, it SHOULD be set to 0, disabling the function. The default value SHOULD be 16.
Beacons supported	Boolean	TRUE or FALSE	A Boolean indicating whether this interface supports beacons
Enhanced Beacons supported	Boolean	TRUE or FALSE	A Boolean indicating whether this interface supports enhanced beacons
ScanType	Boolean	ACTIVE or ENHANCED_ACTIVE	The type of scan to be used when performing a scan for NLME-NETWORK-AND-PARENT-DISCOVERY.request. The ENHANCED_ACTIVE ScanType uses Enhanced Beacons.
InterfaceLinkCostScalar	Integer	1 – 34	This is used to scale all of the Link costs on the interface.. Default is 1.

3.5.2.1 Broadcast Delivery Time

The total delivery time for a broadcast transmission is set to nwkcBroadcastDeliveryTime. This is the amount of time it takes to deliver to all devices in a reasonably large network and was chosen based on empirical analysis of tests that were performed. It is a balance between storing broadcasts for long periods of time and allowing greater throughput of transmitting broadcasts.

3.6 Functional Description

3.6.1 Network and Device Maintenance

All Zigbee devices SHALL provide the following functionality:

- Join a network.
- Leave a network.
- Rejoin a network.

Both Zigbee coordinators and routers SHALL provide the following additional functionality:

- Permit devices to join the network using the following:
    - Association indications from the MAC
    - Explicit join requests from the application
    - Rejoin requests
    - Permit devices to leave the network using the following:
    - Network leave command frames
    - Explicit leave requests from the application
    - Participate in assignment of logical network addresses
    - Maintain a list of neighboring devices

Zigbee coordinators SHALL provide functionality to establish a new network. Zigbee routers and end devices SHALL provide the support of portability within a network.

3.6.1.1 Establishing a New Network

The procedure to establish a new network is initiated through use of the NLME-NETWORK-FORMATION.request primitive. Only devices for which the *nwkcCoordinatorCapable* constant has a value of 0x01, and which are not currently joined to a network SHALL attempt to establish a new network. If this procedure is initiated on any other device, the NLME SHALL terminate the procedure and notify the next higher layer of the illegal request. This is achieved by issuing the NLME-NETWORK-FORMATION.confirm primitive with the Status parameter set to INV_REQUESTTYPE.

When this procedure is initiated, the NLME SHALL first request that the MAC sub-layer(s) perform an energy detection scan over either a specified set of channels or, by default, the complete set of available channels, as dictated by the PHY layer(s) (see [B1]), to search for possible interferers. A channel scan is initiated by issuing an MLME-SCAN.request primitive for each relevant channel mask page to the MAC sub-layer with the ScanType parameter set to energy detection scan. The results are communicated back via the MLME-SCAN.confirm primitive (one for each channel mask page). This scan is not necessary if there is only one channel specified.

On receipt of the results from a successful energy detection scan, the NLME SHALL order the channels on each interface according to increasing energy measurement and discard those channels whose energy levels are beyond an acceptable level. The choice of an acceptable energy level is left to the implementation. The NLME SHALL then perform an active scan, by issuing the MLME-SCAN.request primitive on each MAC Interface with the ScanType parameter set to active scan and ChannelList set to the list of acceptable channels and ChannelPage set to the relevant value for that interface, to search for other Zigbee devices. To determine the best channel on which to establish a new network, the NLME SHALL review the list of returned PAN descriptors and find the first channel for each MAC Interface with the lowest number of existing networks, favoring a channel with no detected networks.

If no suitable channel is found, the NLME SHALL terminate the procedure and notify the next higher layer of the startup failure. This is achieved by issuing the NLME-NETWORK-FORMATION.confirm primitive with the Status parameter set to STARTUP_FAILURE.

If a suitable channel is found, the NLME SHALL select a PAN identifier for the new network. To do this the device SHALL choose a random PAN identifier less than 0xffff that is not already in use on the selected channel. Once the NLME makes its choice, it SHALL set the *macPANID* attribute in the MAC sub-layer to this value by issuing the MLME-SET.request primitive.

If no unique PAN identifier can be chosen, the NLME SHALL terminate the procedure and notify the next higher layer of the startup failure by issuing the NLME-NETWORK-FORMATION.confirm primitive with the Status parameter set to STARTUP_FAILURE.

Once a PAN identifier is selected, the NLME SHALL select a 16-bit network address equal to 0x0000 and set the *nwkNetworkAddress* attribute of the NIB equal to the selected network address.

Once a network address is selected, the NLME SHALL check the value of the *nwkExtendedPANId* attribute of the NIB. If this value is 0x0000000000000000 this attribute is initialized with the value of the MAC constant *aExtendedAddress*.

Once the value of the *nwkExtendedPANId* is checked, the NLME SHALL begin operation of the new PAN by issuing the MLME-START.request primitive to each MAC sub-layer. The parameters of the MLME-START.request primitive SHALL be set according to those passed in the NLME-NETWORK-FORMATION.request, the results of the channel scan, and the chosen PAN identifier. The status of the PAN startup for each MAC Interface is communicated back via the MLME-START.confirm primitive.

On receipt of the status of the PAN startup, the NLME SHALL inform the next higher layer of the status of its request to initialize the Zigbee coordinator. This is achieved by issuing a single NLME-NETWORK-FORMATION.confirm primitive for all MAC Interfaces enabled during the NLME-NETWORK-FORMATION.request. The Status parameter SHALL be set to the value in the primitive returned in the MLME-START.confirm from the MAC sub-layer.

The procedure to successfully start a new network is illustrated in the message sequence chart (MSC) shown in Figure 3-38.

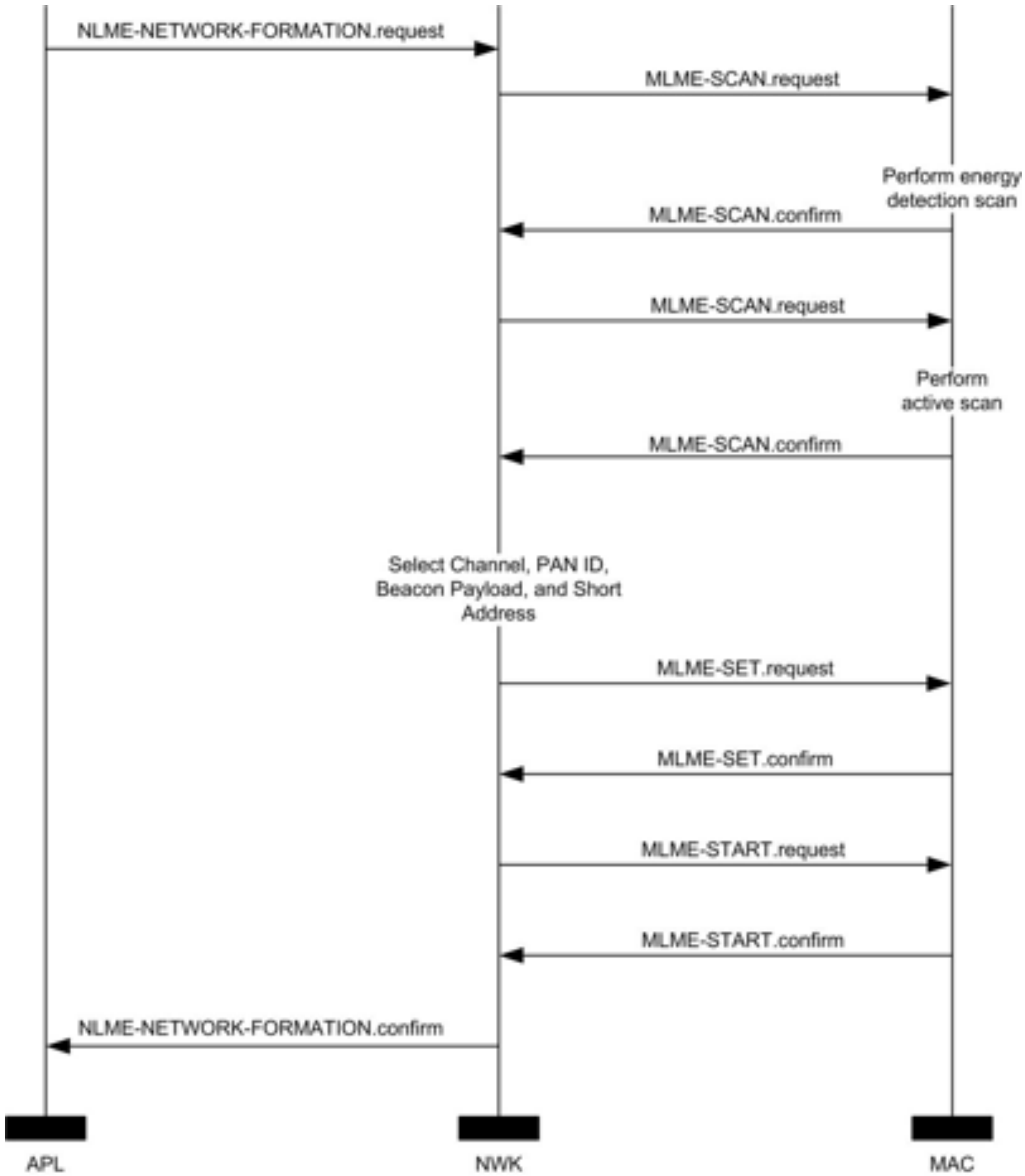


Figure 3-38. Establishing a New Network

3.6.1.2 Permitting Devices to Join a Network

The procedure for permitting devices to join a network is initiated through the NLME-PERMIT-JOINING.request primitive. Only devices that are either the Zigbee coordinator or a Zigbee router SHALL attempt to permit devices to join the network.

When this procedure is initiated with the PermitDuration parameter set to 0x00, the NLME SHALL set the *macAssociationPermit* PIB attribute in the MAC sub-layer to FALSE. A MAC sub-layer attribute setting is initiated by issuing the MLME-SET.request primitive.

When this procedure is initiated with the PermitDuration parameter set to a value between 0x01 and 0xfe, the NLME SHALL set the *macAssociationPermit* PIB attribute in the MAC sub-layer to TRUE. The NLME SHALL then start a timer to expire after the specified duration. On expiration of this timer, the NLME SHALL set the *macAssociationPermit* PIB attribute in the MAC sub-layer to FALSE.

When this procedure is initiated with the PermitDuration parameter set to 0xff, the NLME SHALL set the *macAssociationPermit* PIB attribute in the MAC sub-layer to TRUE for an unlimited amount of time, unless another NLME-PERMIT-JOINING.request primitive is issued.

The procedure for permitting devices to join a network is illustrated in the MSC shown in Figure 3-39.

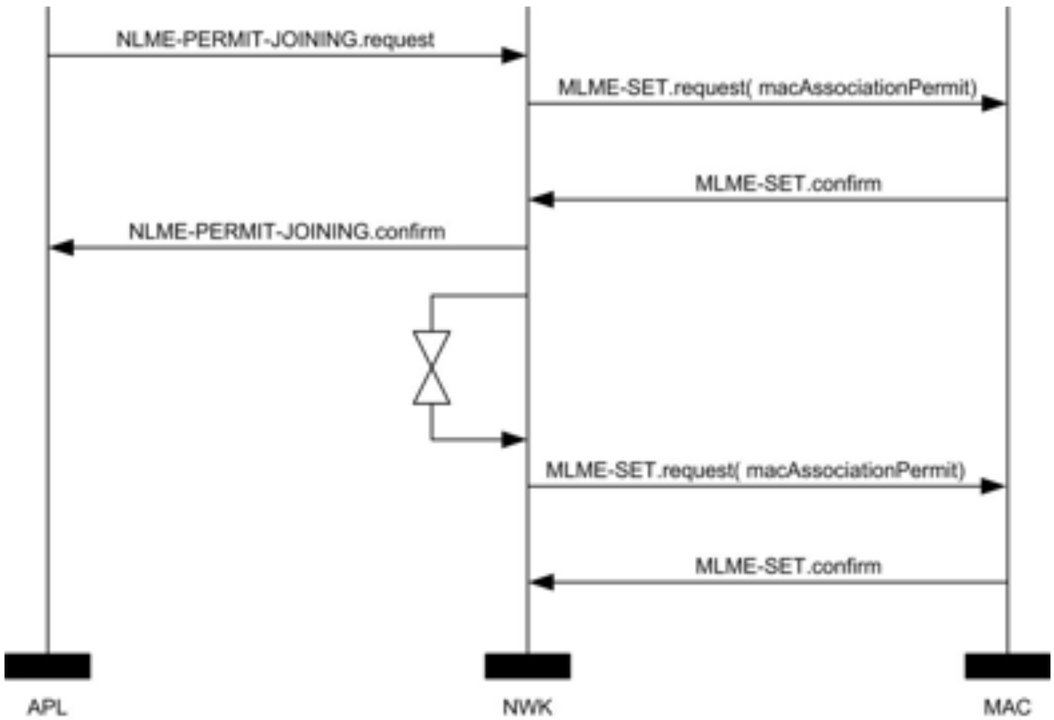


Figure 3-39. Permitting Devices to Join a Network

3.6.1.3 Hub Connectivity

Hub Connectivity is an application layer defined mechanism for detecting the presence of an important application entity, known as a Hub. The mechanism to detect connectivity to it is outside the scope of this specification. The Application has the ability to modify the Hub Connectivity attribute of the core stack so that the core stack advertises this connectivity to the Joining or rejoining devices. Devices SHALL prefer joining or rejoining to Hub Connected routers, but if need be will try routers that do not advertise Hub Connectivity. There MAY be a lag in detection mechanism for Hub connectivity and thus routers advertising they do not have Hub connectivity SHALL still be tried. In Centralized networks the Hub is likely to be the trust center. Distributed networks MAY also make use of hub connectivity even though no trust center is present.

3.6.1.4 Preferred Parent

Preferred Parent is an optional, application defined, mechanism for routers to advertise their capacity to act a parent for another device. Hub Connectivity takes priority over Preferred Parent. When supported, Preferred Parent helps joining devices select between potential parents with the same Hub Connectivity. Details of how routers make this determination is outside the scope of this specification and MAY include next hop LQA to the hub, cost, number of neighbors, and current duty cycle for Sub-GHz devices. Devices that do not make such a determination SHALL always advertise a value of 0. Joining and rejoining devices SHALL prefer parents with a value of 1.

3.6.1.5 Network and Parent Discovery

The NWK layer enables higher layers to discover what networks, if any, are operational in the POS of a device.

The procedure for network discovery SHALL be initiated by issuing the NLME-NETWORK-AND-PARENT-DISCOVERY.request primitive with the ScanChannelsListStructure parameter set to indicate which channels are to be scanned for networks and the ScanDuration parameter set to indicate the length of time to be spent scanning each channel. Upon receipt of this primitive, the NWK layer SHALL issue a MLME-SCAN.request primitives asking each MAC sub-layer to perform an active scan.

When processing and managing beacons for joining or rejoining the network, the device SHALL follow both sections 3.6.1.6 and 3.6.1.6.1.

The Network layer is responsible for filtering beacons based on the parameters passed to the discovery primitive and the device's current state. Beacons that are not suitable for joining or rejoining are discarded. Only beacons representing Zigbee devices that can also act as potential parents are stored in the Discovery Table (*nwkDiscoveryTable*). Potential networks and candidate parents are stored in the Discovery Table.

Once all MAC sub-layer(s) signal the completion of the scan by issuing the MLME-SCAN.confirm primitive to the NLME, the NWK layer SHALL issue the NLME-NETWORK-AND-PARENT-DISCOVERY.confirm primitive containing a description of each network that was heard. Every network description contains the Zigbee version, stack profile, Extended PAN Id, PAN Id, logical channel, and information on whether it is permitting joining.

3.6.1.5.1 Network Discovery

Beacons received during network and parent discovery SHALL be processed as follows.

1) If the MAC beacon has zero length MAC beacon payload, it SHALL be discarded and no further processing SHALL be done.
2) If the ZigbeeVersion is not equal to 0x02 the beacon SHALL be discarded and no further processing SHALL be done.
3) If the StackProfile is not equal to 0x02 the beacon SHALL be discarded and no further processing SHALL be done.
4) If the OnlyPermitJoinNetworks parameter from the NLME-NETWORK-AND-PARENT-DISCOVERY.request was set to TRUE and the MAC beacon indicates PermitJoining is FALSE, the beacon SHALL be discarded and no further processing SHALL be done.
5) If OnlyEndDeviceCapacity parameter from the NLME-NETWORK-AND-PARENT-DISCOVERY.request is set to TRUE and the EndDeviceCapacity is FALSE, the beacon SHALL be discarded and no further processing SHALL be done.
6) If the *nwkExtendedPanId* of the NIB is <u>not</u> equal to 0x0000000000000000 then the following SHALL be done.
   a) The *nwkExtendedPanId* SHALL be compared to the ExtendedPanID field in the beacon.
   b) If the values do not match then the beacon SHALL be discarded and no further processing SHALL be done.
   c) Otherwise continue to step 7.
7) The Beacon data SHALL be added to the Discovery Table described in section 3.6.1.6.1.4.

3.6.1.5.2 Parent Selection

A Zigbee device SHALL maintain an ordered list of possible parents to join or rejoin in the Discovery Table (nwkDiscoveryTable). This list is acquired via NLME-NETWORK-AND-PARENT-DISCOVERY.request. The contents MAY be discarded on generation of NLME-JOIN.confirm. The minimum size of the list is 5. When considering how to rank parents on the list, the following SHALL be considered from highest to lowest priority.

Parents are divided into two categories based on LQA. Any parent with LQA of *nwkGoodParentLQA* or higher is considered good. Any parent with a signal below that is considered marginal. The selection procedure SHALL be evaluated first considering only potential parents with a Good LQA. If and only if this fails to successfully attach this procedure shall be followed a second time, in its entirety, using only potential parents a Marginal LQA. When Active Power Control is used on this link, LQI SHALL be used instead of LQA.

1. A parent that indicates Hub Connectivity is 1 SHALL be preferred over a parent with Hub Connectivity of 0.
2. A parent that indicates a Preferred Parent of 1 SHALL be preferred to a parent with a Preferred Parent of 0.
3. A parent that indicates Long Uptime over a device that indicates Short Uptime.
4. A parent that has the newest NWK Update ID value, considering wrap for an 8-bit value.
5. If the device is rejoining, is an End Device, and the parent is the current parent for the device.
6. Other manufacturer specific feature support that is desired.

When the list becomes full, the lowest rank parent SHALL be dropped for a more favorable parent. The application MAY reorder the potential parents based on higher level knowledge, for example, previous attempts that did not succeed and are not likely to succeed if tried again.

3.6.1.6 Joining or Rejoining a Network

For purposes of the ensuing discussion, a parent-child relationship is formed when a device having membership in the network allows a new device to join. On joining, the new device becomes the child, while the first device becomes the parent.

After joining is complete, a joining or rejoining router device will no longer have a parent child relationship with the device it has joined or rejoined to; rather, both devices will have a relationship of 'sibling' (0x02).

There are many attachment mechanisms to a Zigbee network. When both parent and child support the Network Commissioning commands, the Network Commissioning attach mechanisms SHALL be used. Table 3-66 details what mechanism MAY be used for what operations.

Table 3-66. Zigbee Network Attach Mechanisms

Attach Mechanism	Network Security	Description
MAC Association	None	Join and get network key.
Network Rejoin	None	Rejoin and get network key
Network Rejoin	Encrypted and authenticated	Rejoin, no network key needed
Network Commissioning Commissioning Type = Initial Join	None	Initial join. Trust Center MAY decide whether to negotiate a dynamic link key or simply send the network key with the device's current link key. The network advertises its initial link-key exchange capabilities in Beacon Appendix TLVs. The joiner notifies the trust center of its dynamic link key feature support in TLVs attached to the network commissioning frame. The trust center makes its decision by taking the common subset of supported key negotiation methods into account, preferring the suite with the highest level of security.
Network Commissioning (Trust Center Rejoin) Commissioning Type = Rejoin	None	Rejoin. Trust Center MAY decide whether to negotiate a dynamic link key again or simply send the network with the device's current link key. The network advertises its initial link-key exchange capabilities in Beacon Appendix TLVs. The joiner notifies the trust center of its dynamic link key feature support in TLVs attached to the network commissioning frame. The trust center makes its decision by taking the common subset of supported key negotiation methods into account, preferring the suite with the highest level of security.
Network Commissioning (Secure Rejoin) Commissioning Type = Rejoin	Encrypted and authenticated	Rejoin, no network key needed.

An attach using Network Commissioning with network security and a Commissioning Type of Initial Join is NOT allowed. The diagram in Figure 3-40 shows the decision path for when to select the specific attach mechanism.

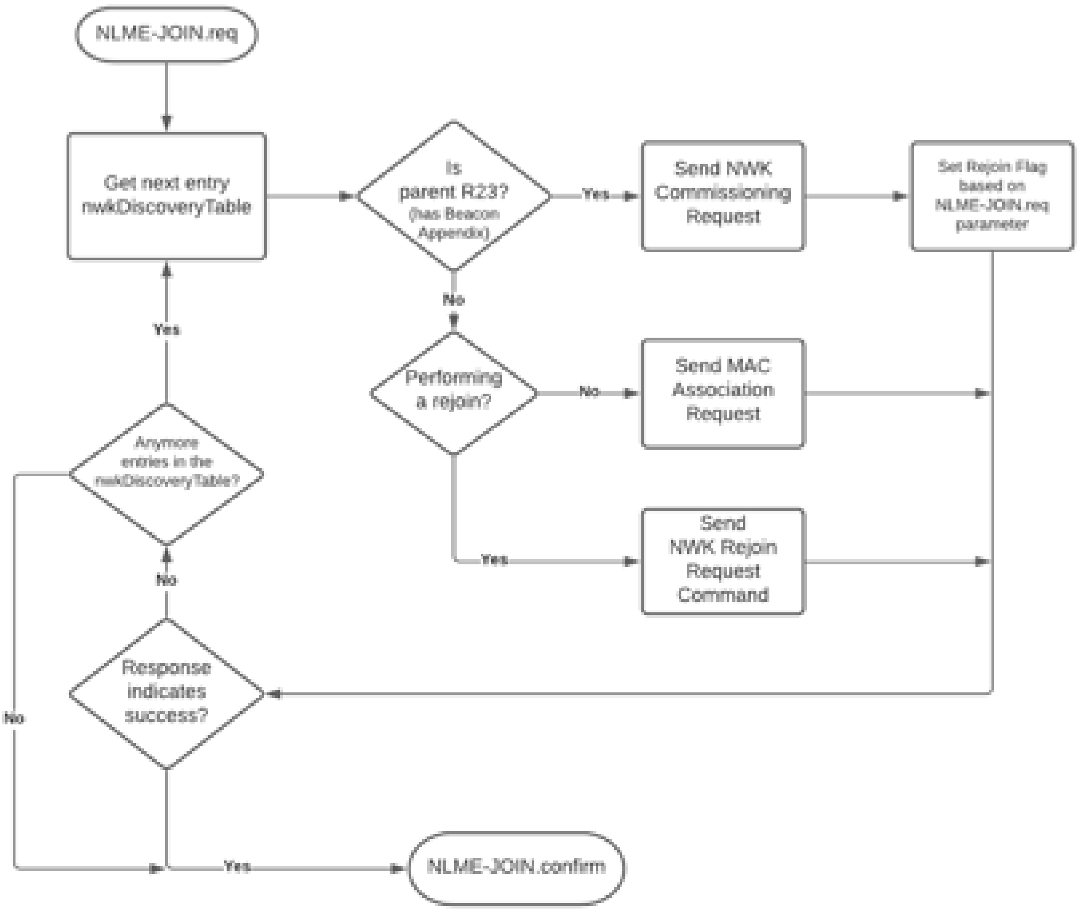


Figure 3-40. Attach Mechanism Decision Tree

3.6.1.6.1 Procedure for Joining or Rejoining a Network

This section specifies the procedure a device (child) SHALL follow if it opts to join or rejoin a network, as well as the procedure a Zigbee coordinator or router (parent) SHALL follow upon receipt of an indication of a device wishing to join or rejoin a network.

3.6.1.6.1.1 Child Procedure

The procedure for joining or rejoining a network SHALL be preceded by network and parent discovery as described in section 3.6.1.5. The next higher layer SHALL wait for receipt of the NLME-NETWORK-AND-PARENT-DISCOVERY.confirm primitive before starting the joining or rejoining process.

If the device is joining, the next higher layer SHALL either choose a network to join from the discovered networks or redo the network discovery. Once a network is selected, it SHALL then issue the NLME-JOIN.request with the RejoinNetwork parameter set to 0x00 and the JoinAsRouter parameter set to indicate whether the device wants to join as a routing device.

Only those devices that are not already joined to a network SHALL initiate the join procedure. If an NLME-JOIN.request is received with RejoinNetwork = 0x00 and the device is already joined, the NLME SHALL terminate the procedure and notify the next higher layer of the illegal request by issuing the NLME-JOIN.confirm primitive with the Status parameter set to INV_REQUESTTYPE.

For both joining and rejoining the NLME-JOIN.request primitive SHALL cause the NWK layer to search its discovery table (nwkDiscoveryTable) for a suitable parent device.

The criteria for determining what order to try potential parents is described in section 3.6.1.5.2.

If the discovery table contains no devices that are suitable parents, the NLME SHALL respond with an NLME-JOIN.confirm with a Status parameter of NOT_PERMITTED. If the discovery table has more than one device that could be a suitable parent the order in which parent devices are tried SHALL follow the rules described in section 3.6.1.5.2.

Once a suitable parent is identified the device SHALL set its *nwkParentInformation* value in the NIB to 0.

The network attach mechanism used to join or rejoin to the network will vary based on the discovery table and the NLME-JOIN.request parameters. The network layer SHALL do the following to determine the attach mechanism.

1) If the potential parent has Beacon Appendix in the nwkDiscoveryTable that is indicative of an R23 or later device, the network layer SHALL do the following.
    a) Network Commissioning SHALL be used as the attach mechanism.
        i) If the RejoinNetwork parameter is set to TRUE then the device SHALL set Commission Type to 0x01, Rejoin.
            (1) If *SecurityEnable* parameter is set to TRUE then the message SHALL be encrypted at the Network Layer with the current network key.
        ii) Else, the Commission type SHALL be set to 0x00, Initial Join with Key Negotiation.
            (1) The *SecurityEnable* parameter SHALL be ignored
2) Otherwise if the RejoinNetwork is set to TRUE, the network layer SHALL use Network Rejoin Request as the attach mechanism.
    a) If Security is set to TRUE then the message SHALL be encrypted at the Network Layer with the current network key.
3) Else, MAC Association SHALL be used as the attach mechanism via the MLME-ASSOCIATE.request primitive.

If MAC Association is used as the attach mechanism, the NLME SHALL issue an MLME-ASSOCIATE.request primitive to the MAC sub-layer and the LogicalChannel parameter of the MLME-ASSOCIATE.request primitive SHALL be set to that found in the discovery table entry corresponding to the coordinator address of the potential parent.

When using Network Rejoin as the attach mechanism, the NLME SHALL issue a Network Rejoin Request Command frame and security SHALL be applied to the command frame if *SecurityEnable* parameter of the NLME-JOIN.request is set to TRUE. It SHALL then follow the procedure in section 3.6.1.6.1.2.

When using Network Commissioning as the attach mechanism, the NLME SHALL issue a Network Commissioning Request and security SHALL be applied to the command frame if SecurityEnable parameter of the NLME-JOIN.request is set to TRUE. It SHALL then follow the procedure in section 3.6.1.6.1.2.

For all attach methods, the bit-fields of the CapabilityInformation parameter SHALL have the values shown in Table 3-67 and the capability information SHALL be stored as the value of the *nwkCapabilityInformation* NIB attribute (see Table 3-62).

Table 3-67. Capability Information Bit-Fields

Bit	Name	Description
0	Alternate PAN coordinator	This field will always have a value of 0 in implementations of this specification.
1	Device type	This field will have a value of 1 if the joining device is a Zigbee router. It will have a value of 0 if the device is a Zigbee end device or else a router-capable device that is joining as an end device.
2	Power source	This field will be set to the value of lowest-order bit of the PowerSource parameter passed to the NLME-JOIN-request primitive. The values are: 0x01 = Mains-powered device 0x00 = other power source
3	Receiver on when idle	This field will be set to the value of the lowest-order bit of the RxOnWhenIdle parameter passed to the NLME-JOIN.request primitive. 0x01 = The receiver is enabled when the device is idle 0x00 = The receiver MAY be disabled when the device is idle
4 – 5	Reserved	This field will always have a value of 0 in implementations of this specification.
6	Security capability	This field SHALL have a value of 0. Note that this overrides the default meaning specified in [B1].
7	Allocate address	This field will have a value of 1 in implementations of this specification.

For joining, if the JoinAsRouter parameter is set to TRUE, the device will function as a Zigbee router in the network. If the JoinAsRouter parameter is FALSE, then it will join as an end device and not participate in routing.

The addressing parameters in the MLME-ASSOCIATE.request primitive (see Chapter 2) SHALL be set to contain the addressing information for the device chosen from the discovery table. The status of the association is communicated back to the NLME via the MLME-ASSOCIATE.confirm primitive.

The result of the attempt to join or rejoin will be reported to the NWK Layer via one of the following means:

- Status parameter of the MLME-ASSOCIATE.confirm
- Status code from the NWK Rejoin Response command
- Status code from the NWK Commissioning Response command
- Timeout

If the attempt to join or rejoin was unsuccessful, the NLME SHALL pick the next potential parent in the ordered list from the discovery table. It SHALL repeat the procedure described in this section until any of the criteria is met:

1. A device is joining and has made a total of nwkMaxInitialJoinParentAttempts.

2. A device is rejoining and has made a total of nwkMaxRejoinParentAttempts.

3. There are no more potential parents for the specified network.

When one of the criteria is met the NLME SHALL then terminate the procedure and issue the NLME-JOIN.confirm primitive with the Status parameter set to the corresponding reason for the failure

If RejoinNetwork is 0x00, or RejoinNetwork is 0x02 and SecurityEnable is FALSE, the device SHALL wait for security data to be received to become fully authenticated on the Zigbee network. The device SHALL set the SecurityTimer for the corresponding nwkNeighborTable entry of its parent to *apsSecurityTimeOutPeriod*. It SHALL decrement the SecurityTimer for every second that has passed before giving up. If the NIB is not updated with the NWK security key then the device SHALL issue an NLME-JOIN.confirm primitive with a status of NO_KEY. Note that the higher layer MAY reset the SecurityTimer due to higher level application messages received.

If the attempt to join or rejoin was successful, the NWK SHALL issue the NLME-JOIN.confirm primitive with a status value of SUCCESS. In this case, the response of the MAC association, NWK Rejoin, or NWK Commissioning SHALL contain a 16-bit logical address unique to that network which the child can use in future transmissions. The NWK layer SHALL then set the Relationship field in the corresponding neighbor table entry to indicate that the neighbor is its parent. By this time, the parent SHALL have added the new device to its neighbor table. Furthermore, the NWK layer will update the values of *nwkNetworkAddress*, *nwkUpdateId* and *mwkPANId* in the NIB.

Once the device has successfully joined the network, if it is a router and the next higher layer has issued a NLME-START-ROUTER.request, the NWK layer SHALL issue the MLME-START.request primitive to its MAC sub-layer. The PANId, LogicalChannel, BeaconOrder and SuperframeOrder parameters SHALL be set equal to the corresponding values held in the neighbor table entry for its parent. The network depth is set to one more than the parent network depth unless the parent network depth has a value of 0x0f, *i.e.* the maximum value for the 4-bit device depth field in the beacon payload. In this case, the network depth SHALL also be set to 0x0f. The PANCoordinator and CoordRealignment parameters SHALL both be set to FALSE. Upon receipt of the MLME-START.confirm primitive, the NWK layer SHALL issue an NLME-START-ROUTER.confirm primitive with the same status value.

Figure 3-41 shows the procedure for a device to join to a network where the parent supports Revision 23 or later of this specification. In this case the MCPS-DATA.request is used to transmit a NWK Commissioning Request command and the MCPS-DATA.indication will contain a NWK Commissioning Response.

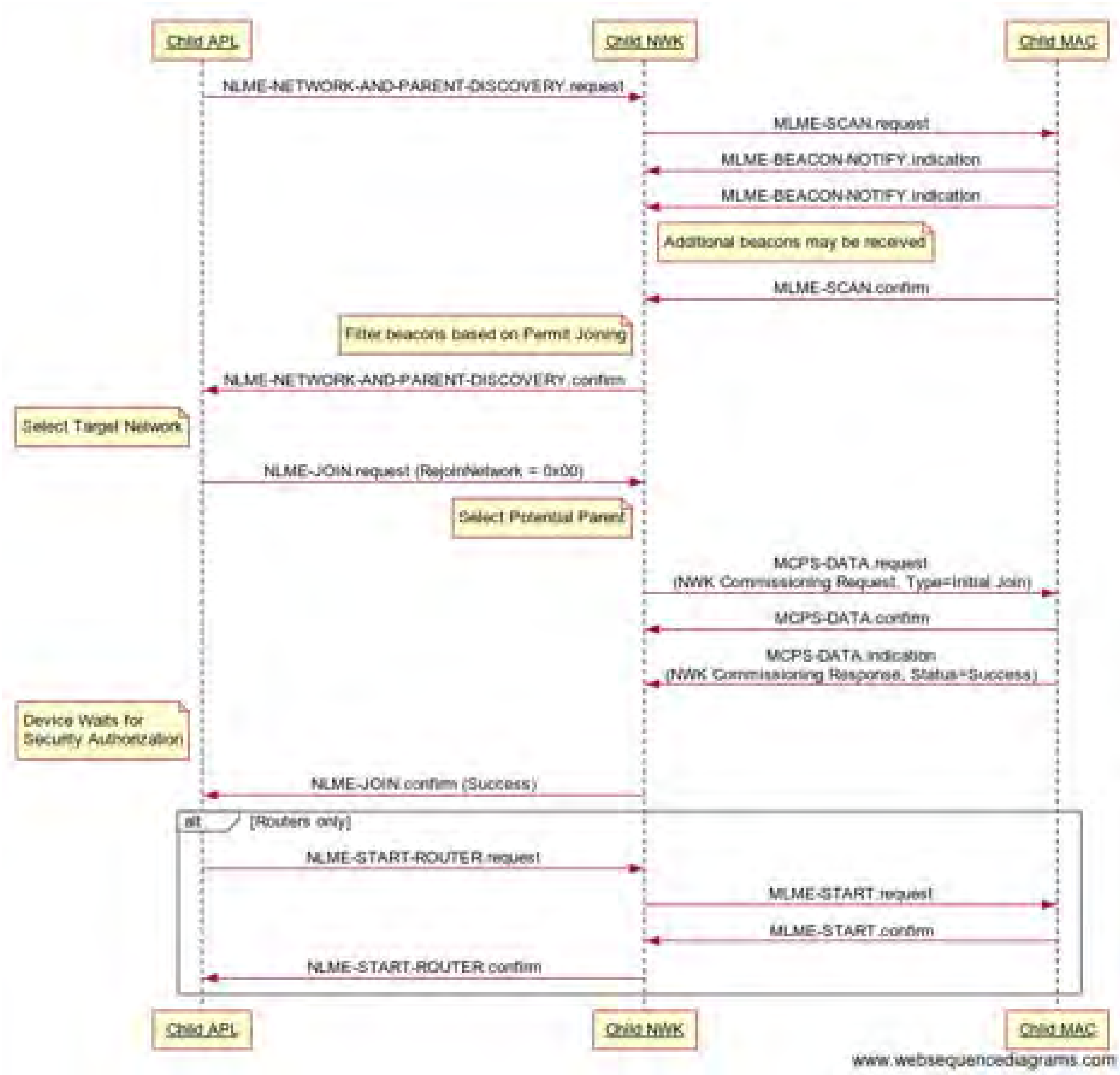


Figure 3-41. Procedure for Joining to a Network (R23+ Parent)

Figure 3-42 shows the procedure for a device joining to a network where the parent supports a version of the specification prior to Revision 23. In this case the MLME-ASSOCIATE primitives are used to join.

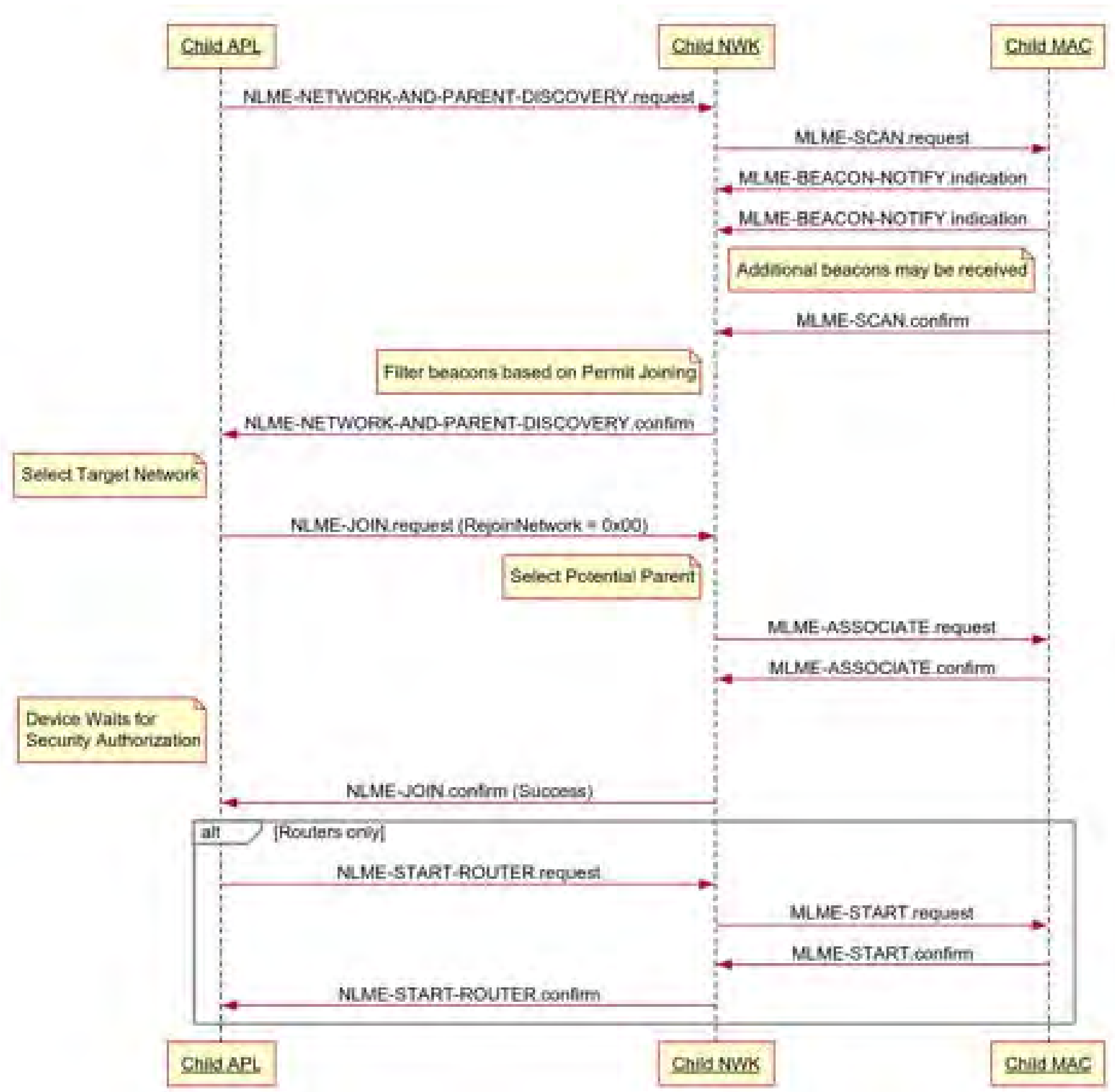


Figure 3-42. Procedure for Joining a Network with a Legacy Parent (Pre-R23)

3.6.1.6.1.2 Transmission and Reception of the NWK Rejoin Request and NWK Commissioning Request

After the successful transmission of the network rejoin request or network commissioning request command using the MAC data service, the network layer SHALL load a countdown timer with a value of *macResponseWaitTime* ([B1]). If the receiver on when idle field of the Capability Information parameter is equal to 0, the device SHALL issue at least one MLME-POLL.request to the potential parent to retrieve the response command before the timer expires. If the receiver on when idle field is equal to 1, polling is not required. Polling more than once before *macResponseWaitTime* ([B1]) elapses is permitted.

If this timer elapses before the appropriate response command frame is received, then the attachment mechanism was unsuccessful.

On receipt of the appropriate response command frame, after the above procedure or at any other time, the device SHALL check the destination IEEE address field and the source IEEE address fields of the command frame NWK header. If the destination IEEE address field is not equal in value to the IEEE address of the receiving device or if the source IEEE address field is not equal in value to the IEEE address of the most recent potential parent to which a rejoin request command frame was sent (or the current parent in the case of an unsolicited rejoin response), then the rejoin response command frame shall be discarded without further processing.

Figure 3-43 shows the procedure for rejoining to a parent that supports Revision 23 or later of the specification. The Network commissioning request is used for the attach mechanism.

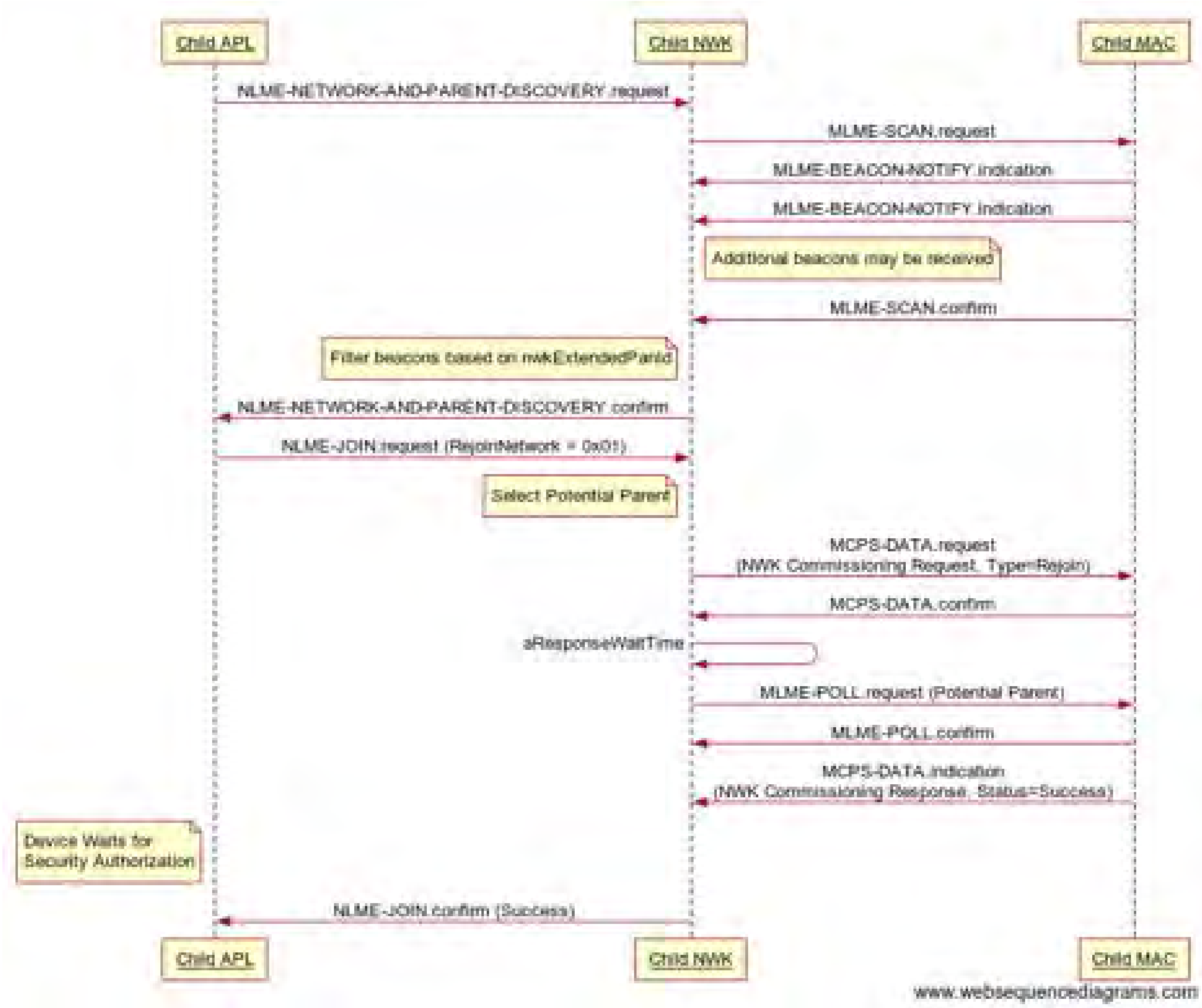


Figure 3-43. Procedure for Rejoining to a Network with a Parent that Supports R23+

Figure 3-44 shows the procedure for rejoining to the network when the parent supports a version prior to Revision 23. The NWK Rejoin Request command is used for the attach mechanism.

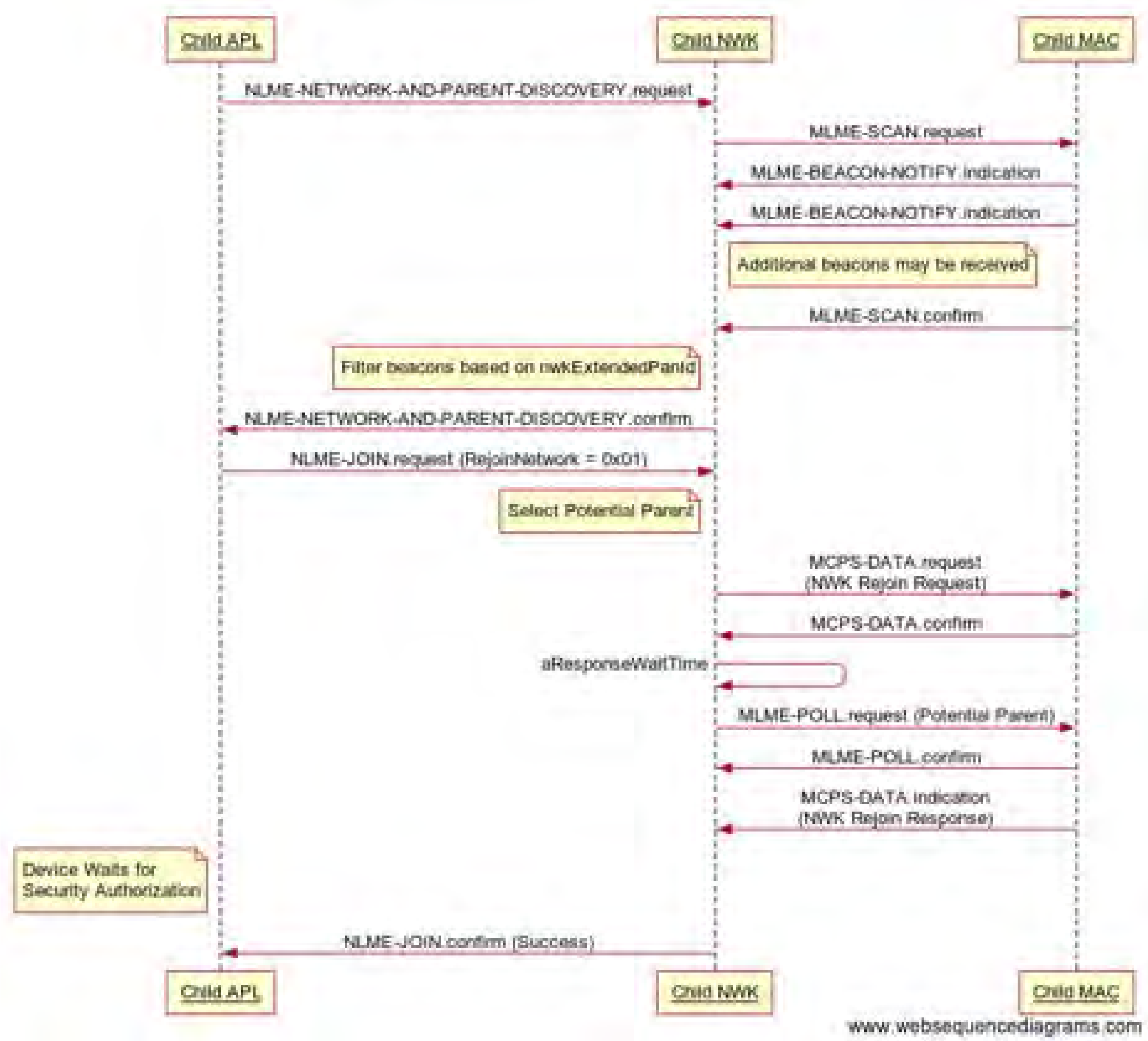


Figure 3-44. Child Rejoining the Network to a Legacy Parent (Pre-R23)

3.6.1.6.1.3 Parent Procedure

The procedure for a Zigbee coordinator or router to allow a device join or rejoin to its network using one of the following mechanisms described in the attach mechanism in Table 3-68.

Table 3-68. Incoming Attach and Response Mechanisms

Incoming Mechanism	Response Method
MLME-ASSOCIATE.indication	MLME-ASSOCIATE.response
NWK Rejoin Request Command Frame	NWK Rejoin Response Command Frame
NWK Commissioning Request Command Frame	NWK Commissioning Response Command Frame

Only those devices that are either a Zigbee coordinator or a Zigbee router and that are permitting devices to join the network SHALL initiate this procedure. If this procedure is initiated on any other device, the NLME SHALL terminate the procedure.

If the parent receives a MLME-ASSOCIATE.indication primitive and the MAC PIB attribute *macAssociationPermit* is FALSE it SHALL issue MLME-ASSOCIATE.response with a status of 0x02, PAN_ACCESS_DENIED.

If the parent receives a NWK Commissioning Request from the NWK layer, and the Commissioning Type is set to 0x00 (Initial Join), and the MAC PIB attribute *macAssociationPermit* is FALSE, it SHALL issue a Network Commissioning Response with a status of 0x02, PAN ACCESS DENIED and halt processing. Otherwise it SHALL continue processing.

If the NWK Commissioning Request command is NWK encrypted and the Commissioning Type indicates Initial Join, the request SHALL be dropped and no further processing SHALL be done.

When this procedure is initiated, , regardless of the incoming mechanism, the NLME of the local device acting as a potential parent SHALL determine whether the device wishing to join or rejoin already exists on its network. To do this, the NLME SHALL search its neighbor table in order to determine whether a matching 64-bit, extended address can be found. If an extended address match is found, the NLME SHALL follow the rules for matching device information below. If no match is found the NLME will continue to process the request.

A rejoin attempt can be received at any time by a parent router. A NWK Rejoin command without security, known as a Trust Center Rejoin, needs to be handled carefully by the stack to prevent state changes on the parent. Unsecured Packets at the network layer claiming to be from existing neighbors (coordinators, routers or end devices) must not rewrite legitimate data in the nwkNeighborTable.

If the NWK Rejoin Request command frame received at the parent router does not have network layer encryption, the parent router SHALL look at the apsTrustCenterAddress in the AIB. If the value of apsTrustCenterAddress is 0xFFFFFFFFFFFFFFFF, the rejoin attempt SHALL be rejected. The Status parameter of the NWK Rejoin Response command shall indicate PAN ACCESS DENIED.

If the value of apsTrustCenterAddress is NOT 0xFFFFFFFFFFFFFFFF and an existing, matching entry in the nwkNeighborTable has been found, then the rules for matching device information SHALL be applied as follows:

1. Compare the following: the Network Address of the nwkNeighborTable to the NWK Short Address of the command, the Device Type enum of the nwkNeighborTable to the Device type bit of the MAC Capabilities in the command, and the RxOnWhenIdle bit to the RxOnWhenIdle bit of the MAC Capabilities in the command.
2. If the NWK Rejoin Request command frame has network layer encryption that passes security processing in section 4.3.1.2, then NLME shall consider the rejoin attempt successful. Any values that changed in step 1 can be updated in the nwkNeighborTable.
3. If the NWK Rejoin Request command frame does not have network layer encryption and the NWK short address and/or capabilities values are different but all other values in step 1 are the same including the 64 bit extended address, then the NLME SHALL reject such a rejoin attempt. It SHALL send a rejoin response with PAN ACCESS DENIED. No changes SHALL be made to the existing nwkNeighborTable entry corresponding to the attempted rejoin.

If no match to an existing device in the nwkNeighborTable was found and the potential parent does not have the capacity to accept more children, the NLME SHALL terminate the procedure and indicate this fact in the subsequent response mechanism described in Figure 3-18. The Status parameter of that primitive or command frame SHALL indicate that the PAN is at capacity.

If the request to join or rejoin is granted by the parent, the NLME of the parent SHALL create a new entry for the child in its neighbor table using the supplied device information and indicate a successful join or rejoin via the response mechanism noted in Figure 3-18. The relationship field of the new neighbor table entry SHALL be set to the value 0x01 only if the mechanism was NWK Rejoin and had NWK Layer security. Otherwise, the relationship field SHALL be set to 0x05 indicating an unauthenticated child. The status of the response transmission to the child is communicated back to the network layer via the MLME-COMM-STATUS.indication primitive.

When a new entry for the child has been created in the parent's neighbor table, the parent SHALL search its routing table for an entry where the destination address equals the address of the rejoining device and delete such entry, if one exists.

If the transmission was unsuccessful (*i.e.* the MLME-COMM-STATUS.indication primitive contained a Status parameter not equal to SUCCESS), the NLME SHALL terminate the procedure. If the transmission was successful, the NLME SHALL notify the next higher layer that a child has just joined the network by issuing the NLME-JOIN.indication primitive.

Authorization onto the Zigbee network is not performed by the NLME. The NLME on the parent SHALL result in an APSME-UPDATE-DEVICE.request issued locally to the router to request the Trust Center authorize or reject the new device. . If the Joiner Encapsulation Global TLV is present it SHALL be passed directly to the JoiningDeviceTLVs parameter of the APSME-UPDATE-DEVICE.request interface. For the TLVs received by the NLME-JOIN.indication, only the Joiner Encapsulation Global TLV SHALL be passed to the APSME-UPDATE-DEVICE.request, all other TLVs shall be processed by the parent locally by the NLME. The JoinerMethod of the NLME-JOIN.indication request SHALL be mapped to the APSME-UPDATE-DEVICE.request per Table 3-69.

Table 3-69. NLME-JOIN.indication JoinerMethod and APSME-UPDATE-DEVICE.request status mapping

JoinerMethod of NLME-JOIN.indication	Status of APSME-DEVICE-UPDATE.request
0x00 = MAC Association	0x01 = Standard Device Unsecured Join
0x01 = Network Rejoin without Security	0x03 = Standard Device Trust Center Rejoin
0x02 = Secured Network Rejoin	0x00 = Standard Device Secured Rejoin
0x03 = Network Commissioning Join without Security	0x01 = Standard Device Unsecured Join
0x04 = Network Commissioning Rejoin without Security	0x03 = Standard Device Trust Center Rejoin
0x05 = Secure Network Commissioning Rejoin	0x00 = Standard Device Secured Rejoin

The parent SHALL set the SecurityTimer for the corresponding neighbor table entry to *apsSecurityTimeOutPeriod*. The parent will wait to receive any network encrypted message from the device indicating it has been authorized onto the network. If the SecurityTimer has reached zero and no message is received the parent SHALL delete the unauthenticated child from its neighbor table. Note that the higher layer of the parent can reset the SecurityTimer value for the neighbor table entry as higher layer messages are relayed to the device indicating it is still being authorized for joining or rejoining to the network.

The procedure for successfully joining a device to the network via MAC Association is illustrated in the message sequence chart shown in Figure 3-45.

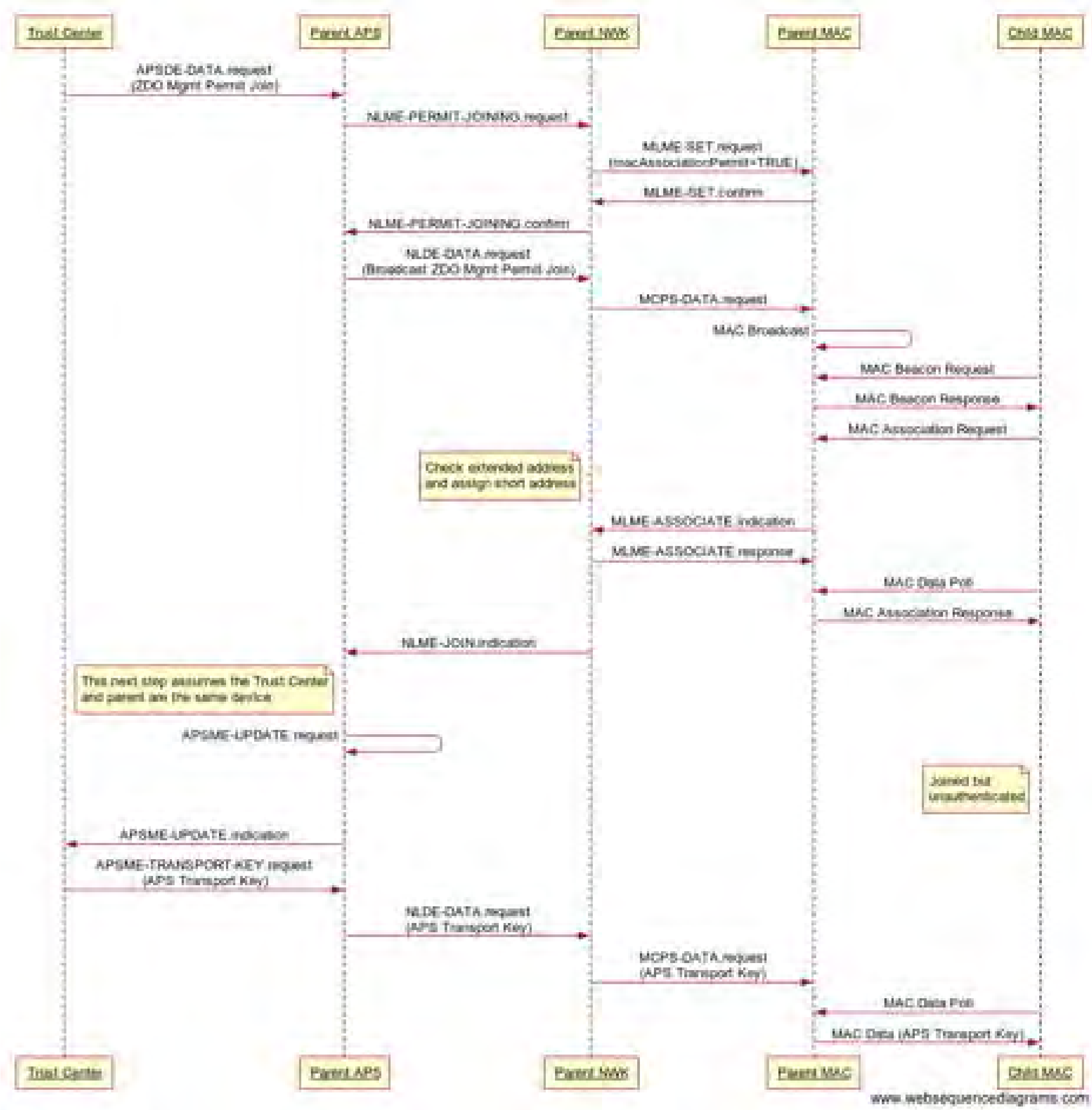
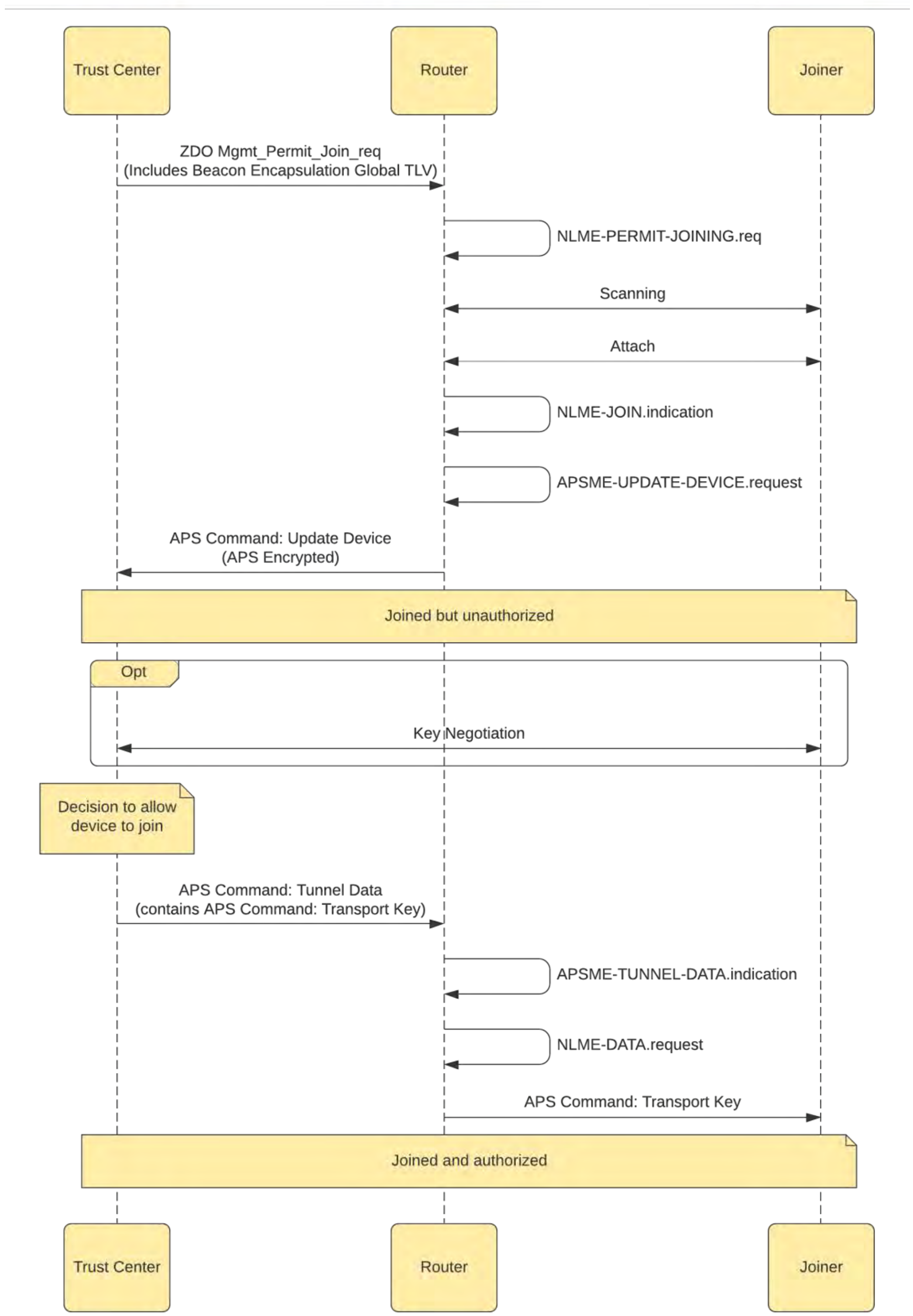


Figure 3-45. Parent Process for Receiving a Request to Join Using MAC Association

The procedure for successfully rejoining a device to the network via NWK Rejoin is illustrated in the message sequence chart shown in .



8031

8032 **Figure 3-46. Parent Processing for receiving a request to rejoin via NWK Rejoin Command**

8033 The procedure for successfully joining or rejoining a device to the network via NWK Commissioning is illustrated in
8034 the message sequence chart shown in Figure 3-47.

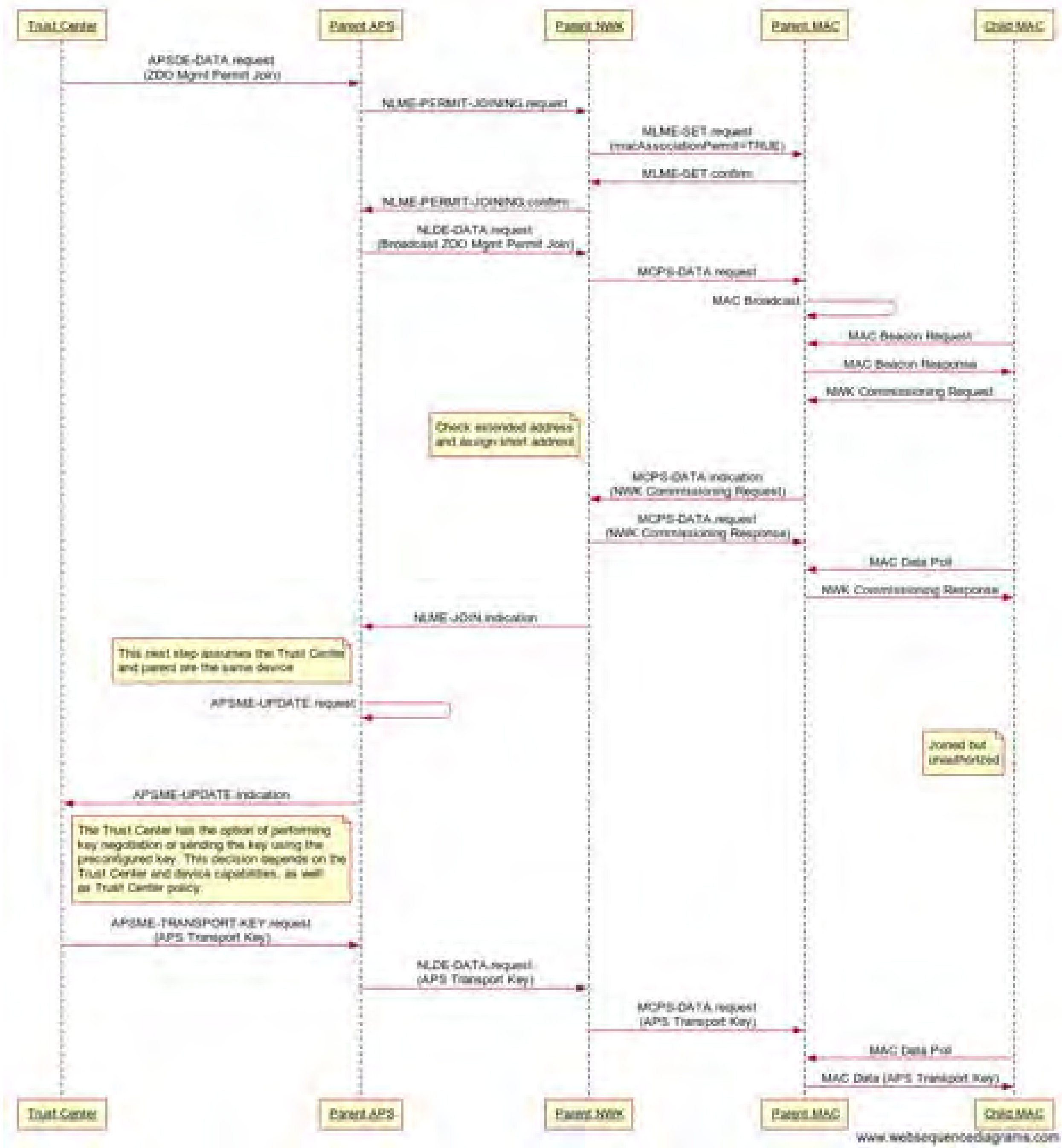


Figure 3-47. Parent Procedure for Processing a Received Join or Rejoin via NWK Commissioning

3.6.1.6.1.4 Discovery Table

The stack SHALL maintain a separate table for storing potential networks and parents during join and rejoin operations. The minimum size of this table is 6 entries. This table is described in Table 3-70.

Table 3-70. Discovery Table Fields (nwkDiscoveryTable)

Field Name	Field Type	Valid Range	Description
Extended PAN ID	Integer	Any	The 64-bit unique identifier of the network to which the device belongs to.
Logical Channel	Integer	Selecting from the available logical channels support by the PHY	The logical channel on which the network is operating.
Parent Priority	Integer	1 – nwkDiscoveryTableSize	This is the priority that the potential parents SHALL be considered for join or rejoin attempts. 1 is considered the highest priority and 10 is the lowest.
Short ID	Integer	0x0000 – 0xFFFF	The short ID of the potential parent.
LQA (LQI)	Integer	0 – 255	The LQA (LQI) value of the potential parent.
Update ID	Integer	0x00 – 0xFF	The value of the NWK Update ID of the beacon.
Beacon Appendix TLVs	Array of bytes	Any	A set of TLVs indicating information about the network.

3.6.1.7 Neighbor Tables

The neighbor table of a device SHALL contain information on every device on the current Zigbee network within transmission range, up to some implementation-dependent limit.

The neighbor does not store information about potential networks and candidate parents to join or rejoin. The Discovery table SHALL be used for this.

The neighbor table is used to store information about neighbors, whether they are fully authorized in the network or are in the process of joining or rejoining. After the device has been authorized on a network, it is used to store relationship and link-state information about neighboring devices in that network. A table entry SHALL be updated every time a device receives any frame from the corresponding neighbor.

The outgoing cost field contains the cost of the link as measured by the neighbor. The value is obtained from the most recent link status command frame received from the neighbor. A value of 0 indicates that no link status command listing this device has been received.

The age field indicates the number of *nwkLinkStatusPeriod* intervals that have passed since the last link status command frame was received, up to a maximum value of *nwkRouterAgeLimit*.

Mandatory and optional data that are used in normal network operation are listed in Table 3-71.

Table 3-71. Neighbor Table Entry Format (nwkNeighborTable)

Field Name	Field Type	Valid Range	Description
Extended address	Integer	An extended 64-bit, IEEE address	64-bit IEEE address that is unique to every device.
Network address	Network address	0x0000 – 0xfff7	The 16-bit network address of the neighboring device. This field SHALL be present in every neighbor table entry.
Device type	Integer	0x00 – 0x02	The type of neighbor device: 0x00 = Zigbee coordinator 0x01 = Zigbee router 0x02 = Zigbee end device This field SHALL be present in every neighbor table entry.
RxOnWhenIdle	Boolean	TRUE or FALSE	Indicates if neighbor's receiver is enabled during idle periods: TRUE = Receiver is on FALSE = Receiver is off This field SHOULD be present for entries that record the parent or children of a Zigbee router or Zigbee coordinator.
End Device Configuration	Bitmask	0x0000 – 0xFFFF	The end device's configuration. See section 3.4.11.3.2. The default value SHALL be 0.
Timeout Counter	Integer	0x00000000 – 0x00F00000	This field indicates the current time remaining, in seconds, for the end device.
Device Timeout	Integer	0x00000000 – 0x0001FA40	This field indicates the timeout, in seconds, for the end device child. The default value for end device entries is calculated by using the *nwkEndDeviceTimeoutDefault* value and indexing into Table 3-54, then converting the value to seconds. End Devices MAY negotiate a longer or shorter time using the NWK Command End Device Timeout Request.

Field Name	Field Type	Valid Range	Description
Relationship	Integer	0x00 – 0x09	The relationship between the neighbor and the current device: 0x00=neighbor is the parent 0x01=neighbor is a child 0x02=neighbor is a sibling 0x03=none of the above 0x04=previous child 0x05=unauthenticated child 0x06=unauthorized child with relay allowed 0x07=neighbor is a lost child 0x08=neighbor is a child with address conflict 0x09=neighbor is a backbone mesh sibling This field SHALL be present in every neighbor table entry.
Transmit Failure	Integer	0x00 – 0xff	A value indicating if previous transmissions to the device were successful or not. Higher values indicate more failures. This field SHALL be present in every neighbor table entry.
LQA (LQI)	Integer	0x00 – 0xff	The estimated link quality for RF transmissions from this device. See section 3.6.4.1 for a discussion of how this is calculated. This field SHALL be present in every neighbor table entry.
Outgoing Cost	Integer	0x00 – 0xff	The cost of an outgoing link as measured by the neighbor. A value of 0 indicates no outgoing cost is available. This field is mandatory.
Age	Integer	0x00 – 0xff	The number of nwkLinkStatusPeriod intervals since a link status command was received. This field is mandatory.

Field Name	Field Type	Valid Range	Description
Incoming beacon timestamp	Integer	0x000000 – 0xffffff	The time, in symbols, at which the last beacon frame was received from the neighbor. This value is equal to the timestamp taken when the beacon frame was received, as described in IEEE Std 802.15.4-2020 [B1]. This field is optional.
Beacon transmission time offset	Integer	0x000000 – 0xffffff	The transmission time difference, in symbols, between the neighbor's beacon and its parent's beacon. This difference MAY be subtracted from the corresponding incoming beacon timestamp to calculate the beacon transmission time of the neighbor's parent. This field is optional.
Keepalive Received	Boolean	TRUE or FALSE	This value indicates at least one keepalive has been received from the end device since the router has rebooted.
MAC Interface Index	Integer	0 – 31	This is an index into the MAC Interface Table indicating what interface the neighbor or child is bound to.
MacUnicastBytesTransmitted	Integer	0 – 4,294,967,296	The number of bytes transmitted via MAC unicast to the neighbor. This is an optional field.
MacUnicastBytesReceived	Integer	0 – 4,294,967,296	The number of bytes received via MAC unicasts from this neighbor. This is an optional field.
RouterAge	Integer	0x0000 - 0xffff	The number of nwkLinkStatusPeriod intervals, which elapsed since this router neighbor was added to the neighbor table. This value is only maintained on routers and the coordinator and is only valid for entries with a relationship of 'parent', 'sibling' or 'backbone mesh sibling'. This is a saturating up-counter, which does not roll-over.

Field Name	Field Type	Valid Range	Description
RouterConnectivity	Integer	0x00 - 0xb6	An indicator for how well this router neighbor is connected to other routers in its vicinity. Higher numbers indicate better connectivity. This metric takes the number of mesh links and their incoming and outgoing costs into account. This value is only maintained on routers and the coordinator and is only valid for entries with a relationship of 'parent', 'sibling' or 'backbone mesh sibling'.
RouterNeighborSetDiversity	Integer	0x00 - 0xff	An indicator for how different the sibling router's set of neighbors is compared to the local router's set of neighbors. Higher numbers indicate a higher degree of diversity. This value is only maintained on routers and the coordinator and is only valid for entries with a relationship of 'parent', 'sibling' or 'backbone mesh sibling'.
RouterOutboundActivity	Integer	0x00 - 0xff	A saturating counter, which is preloaded with nwkRouterAgeLimit when this neighbor table entry is created; incremented whenever this neighbor is used as a next hop for a data packet; and decremented unconditionally once every nwkLinkStatusPeriod. This value is only maintained on routers and the coordinator and is only valid for entries with a relationship of 'parent', 'sibling' or 'backbone mesh sibling'.
RouterInboundActivity	Integer	0x00 - 0xff	A saturating counter, which is preloaded with nwkRouterAgeLimit when this neighbor table entry is created; incremented whenever the local device is used by this neighbor as a next hop for a data packet; and decremented unconditionally once every nwkLinkStatusPeriod. This value is only maintained on routers and the coordinator and is only valid for entries with a relationship of 'parent', 'sibling' or 'backbone mesh sibling'.

Field Name	Field Type	Valid Range	Description
SecurityTimer	Integer	0 – 255	If the local device is joined to the network this is a countdown timer indicating how long an "unauthorized child" neighbor is allowed to be kept in the neighbor table. If the timer reaches zero the entry SHALL be deleted. If the local device is an unauthorized child and not fully joined to the network, this is a timer indicating how long it will maintain its parent before giving up the join or rejoin. If the timer reaches zero then the device SHALL leave the network.

3.6.1.8 Stochastic Address Assignment Mechanism

Network short addresses SHALL be chosen at random. The random address assigned SHALL conform to the NIST testing regimen described in reference [B10]. When a device joins the network using MAC association, its parent SHALL choose a random address that does not already appear in any entry in the parent's NIB. Under stochastic addressing, once a device has been assigned an address, it has no reason to relinquish that address and SHOULD retain it unless it receives an indication that its address is in conflict with that of another device on the network. Furthermore, devices MAY self-assign random addresses under stochastic addressing and retain them, as in the case of joining a network using the rejoin command frame (see section 3.6.1.6.1.2). The Zigbee coordinator, which has no parent, SHALL always have the address 0x0000.

3.6.1.9 Installation and Addressing

ZigbeeUnder stochastic address assignment, *nwkcMaxDepth* is related to the number of hops across the network, i.e. the maximum number of hops equals 2 * nwkcMaxDepth. This is not a controlled value in networks using stochastic address assignment.

3.6.1.10 Address Conflicts

An address conflict occurs when two devices in the same network have identical values for *nwkNetworkAddress*. Preventing all such conflicts, for example by using tree address assignment and prohibiting the reuse of assigned addresses, is not always practical. This section describes how address conflicts that do occur can be detected and corrected. Address conflict detection SHALL always be enabled.

Note that the network addresses used in routing messages are verified during the route discovery process. The device_annc now is also used to verify addresses. The verification applies only to devices, links, and information present at the time of the discovery or device_annce. Verification can be achieved at other times, such as before sending a unicast directly to a neighbor, by sending a network status command with a status code value of 0x0e, indicating address verification.

If a device receives a broadcast data frame and discovers an address conflict as a result of the receipt, as discussed below in section 3.6.1.10.2, it SHOULD NOT retransmit the frame as usual but SHALL discard it before taking the resolution actions described in section 3.6.1.10.3.

3.6.1.10.1 Obtaining Address Information

The NWK layer obtains address information from incoming messages, including both NWK commands and data messages. Address information from data messages is passed to the NWK layer by being added to the network address map table in the NIB.

The ability to detect address conflicts is enhanced by adding one or both of the Destination IEEE Address and Source IEEE Address fields to a message's NWK frame. All NWK command messages SHALL contain the source IEEE address and also the destination IEEE address if it is known by the source device, unless explicitly stated otherwise.

Route request commands SHALL include the sender's IEEE address in the Sender IEEE address field. This ensures that devices are aware of their neighbors' IEEE addresses.

3.6.1.10.2 Detecting Address Conflicts

Routers and coordinators are responsible for detecting address conflicts before new devices join the network. When answering requests to join a network at the network layer (for example, answering a Network Rejoin Request), parent routers and coordinators SHALL ensure that new device addresses do not conflict within their local address map, *nwkAddressMap*. This will not detect all possible conflicts in the network but is intended to resolve obvious conflicts before the device starts using a conflicting address. If an address conflict is detected the process for resolving is described in section 3.6.1.10.3.

After joining a network or changing address due to a conflict, a ZDO device SHALL send either a device_annc or initiate a route discovery prior to sending messages. This allows all routers and the coordinator in the entire network to detect conflicts by examining a message that contains both the short address and the long address.

Upon receipt of a frame containing a 64-bit IEEE address in the NWK header, the contents of the *nwkAddressMap* attribute of the NIB and neighbor table SHOULD be checked for consistency.

If the destination address field of the NWK Header of the incoming frame is equal to the *nwkNetworkAddress* attribute of the NIB then the NWK layer SHALL check the destination IEEE address field, if present and even if it is the 0xffffffffffffffff address, against the value of *aExtendedAddress*. If the IEEE addresses are not identical then a local address conflict has been detected on *nwkNetworkAddress*.

If a neighbor table or address map entry is located in which the 64-bit address is the null IEEE address (0x00....00), the 64-bit address in the table can be updated. However, if the 64-bit address is not the null IEEE address and does. not correspond to the received 64-bit address, the device has detected a conflict elsewhere in the network.

A new address map entry SHALL be added if all of the following are true:

1. No address conflict was detected.
2. No entry for neither the short nor long address exists.
3. There is space in the nwkAddressMap table.

When a broadcast frame is received that creates a new BTR, if the Source Address field in the NWK Header is equal to the *nwkNetworkAddress* attribute of the NIB then a local address conflict has been detected on *nwkNetworkAddress*. This SHALL only apply if the broadcast is received after nwkNetworkBroadcastDeliveryTime of being powered up and operating on the network. If the device has been operating on the network for less than nwkNetworkBroadcastDeliveryTime, it SHALL NOT trigger an address conflict.

Address conflicts are resolved as described in section 3.6.1.10.3.

3.6.1.10.3 Resolving Address Conflicts before a Device Joins or Rejoins

During the joining process, parent routers and coordinators are expected to resolve address conflicts locally when receiving a join or rejoin request message. Routers and coordinators are expected to assign unique addresses that do not conflict with their local address map. The mechanism for doing so will vary based on the attach mechanism used by the device and is described below.

It is important to note that network wide notification of address conflicts SHALL NOT be generated due to unencrypted messages. This includes the join and rejoining message exchanges. In other words, an unencrypted network frame SHALL NOT generate a Network Status message.

If the joining device uses MAC association request command, then the router or coordinator parent will assign a short address by randomly generating an address and ensuring it does not conflict with its local address map, nwkAddressMap. It uses this value in the Short Address field in the payload of the MAC Association response command frame.

If the device uses NWK Rejoin request command (secured or unsecured) it will use short MAC addressing at the MAC layer and the network header will include both the short and long addresses. The router or coordinator parent SHALL check the short address against its local address map, *nwkAddressMap*. If the device is not allowed to attach, the parent router can simply respond back to the same short address with the appropriate status code, such as Pan at Capacity. If the device is allowed to attach to the parent and no conflict is detected, then the parent SHALL allow the use of that short address and return that same short address in the Network Address field in the payload of the Network Rejoin Response command frame. If the device is allowed to attach but the parent has detected an address conflict, the parent SHALL NOT allow use of that address. The parent SHALL randomly select a new, non-conflicting address and return that field in the Network Address field in the payload of the Network Rejoin Response. The message will still be addressed to the old short address at the MAC layer. The parent SHALL still allow the rejoin to succeed in that case. The device SHALL change its address on receipt of the Network Rejoin Response before sending any more messages on the network.

If the device uses NWK Commissioning Request command (secured or unsecured), the procedure is different than rejoin. This is due to the fact that the Network Commissioning request can contain TLV data that uses the short address that the device wants to use. In the case of short address conflict with the Network Commissioning Request command, the parent will reject the attempt and tell the device to retry. The procedure is described below.

If the device is not allowed to attach, the parent router can simply respond back to the same short address with the appropriate status code, such as Pan at Capacity.

If the parent allows the device to attach with the Network Commissioning Request and the short address does not conflict, then the parent SHALL allow the use of that short address and return that same short address in the Network Address field in the payload of the Network Commissioning Response command frame.

If the parent allows the device to attach with the Network Commissioning Request and the short address conflicts with its local address map, *nwkAddressMap*, the parent SHALL NOT allow the use of that address. The parent SHALL randomly select a new, non-conflicting address and return that field in the Network Address field in the payload of the Network Commissioning Response. The message will still be addressed to the old short address at the MAC layer. The parent SHALL NOT indicate SUCCESS in the status field of the Network Commissioning Response. In that case, it SHALL indicate 0xF0 for the status, Address Conflict. The device is expected to retry the Network Commissioning Request using that new address in all fields that reference a short address.

3.6.1.10.4 Trust Center Swap-out Special Consideration

A Trust Center swap-out is a rare occurrence that is detailed in section 4.7.4. During that event a device may be rejoining directly to the coordinator that has short address 0x0000 but whose long address has changed unbeknownst to the rejoining device. This may result in the short address of network messages being correct but the long address being different than what is expected. As dictated previously, network wide address conflict notifications SHALL NOT be generated due to unencrypted messages at the network layer.

Devices SHALL accept ***unencrypted*** network layer messages that contain both long and short destination addresses where only the short address matches the local nwkNetworkAddress NIB parameter. The processing of these messages is limited based on the contents and the policies dictated in the specification.

Devices SHALL generate address conflict notifications normally when receiving network ***encrypted*** messages as described in section 3.6.1.10.2.

3.6.1.10.5 Resolving Address Conflicts during Normal Operation

If a Zigbee coordinator or Router determines that there are multiple users of an address that is not its own, it SHALL inform the network by broadcasting a network status command with a status code of 0x0d indicating address conflict, and with the offending address in the destination address field. The network status command SHALL be broadcast to 0xFFFD, i.e. all devices with *macRxOnWhenIdle* = TRUE. The device SHALL delay initiation of this broadcast by a random jitter amount bounded by *nwkcMaxBroadcastJitter*. If during this delay a network status is received with the identical payload, the device SHALL cancel its own broadcast.

If the device has learned of the conflict other than receiving a network status command with a status of 0x0d, then it SHALL inform the network by broadcasting a network status command with a status code of 0x0d indicating address conflict, and with its previous address in the destination address field. The network status command SHALL be broadcast to 0xFFFD—that is, all devices with macRxOnWhenIdle= TRUE. The device SHALL delay initiation of this broadcast by a random jitter amount bounded by nwkcMaxBroadcastJitter. If during this delay a network status is received with the identical payload, the device SHALL cancel its own broadcast. Regardless of how it learned of the conflict, it SHALL implement the procedure on detecting address conflicts detailed in section 3.6.1.10.2.

End devices SHALL NOT resolve any conflicts. The conflicts for end devices would be resolved by their parent device.

A Zigbee Coordinator SHALL not change its address.

If a parent device detects or is informed of a conflict with the address of an end device child, the parent SHALL pick a new address for the end device child. If the end device is a non-sleepy then the router SHALL send immediately an unsolicited rejoin response command frame to inform the end device child of the new address. If the end device is a sleepy device it SHALL send the unsolicited rejoin response to inform the end device child of the new address upon receiving the next keepalive message, and this message SHALL take precedence over any other network or application layer message. The unsolicited rejoin response SHALL always use network encryption. To notify the next higher layer of an address change the end device SHALL issue an NLME-NWK-STATUS.indication with status 'Network Address Update' and the new network address as the value of the ShortAddr parameter.

A Zigbee Router or End Device that changes its address SHALL do the following:

1. Send a broadcast ZDO Device_announce_req with its new short address. The device SHALL jitter the message by nwkcMaxBroadcastJitter.

3.6.1.10.6 Security Considerations for Rejoin Response and Network Commissioning Response

The Network Rejoin Response and the Network Commissioning Response can be sent with or without network security depending on the usage as detailed in the specification. The end device needs to check the security of these messages only when the end device explicitly initiates a rejoin request or commissioning request is the response allowed to be unencrypted.

A Network Rejoin Response and the Network Commissioning Response SHALL only be accepted without network encryption when the device has made a corresponding unencrypted Network Rejoin Request or a Network Commissioning Request. Unencrypted, unsolicited Network Rejoin Response and the Network Commissioning Response SHALL be dropped and no further processing SHALL be done.

For further clarity, the unsolicited Network Rejoin Response used when an address conflict is detected SHALL be network encrypted. A device that receives an unsolicited and unencrypted network rejoin response or an unsolicited and unencrypted network commissioning response SHALL drop the message and no further processing SHALL be done.

3.6.1.11 Leaving a Network

This section specifies methods for a device to remove itself from the network and for the parent of a device to request its removal. In both cases, the children of the removed device, if any, MAY also be removed.

3.6.1.11.1 Method for a Device to Initiate Its Own Removal from the Network

This section describes how a device can initiate its own removal from the network in response to the receipt of an NLME-LEAVE.request primitive from the next higher layer as shown in Figure 3-48.

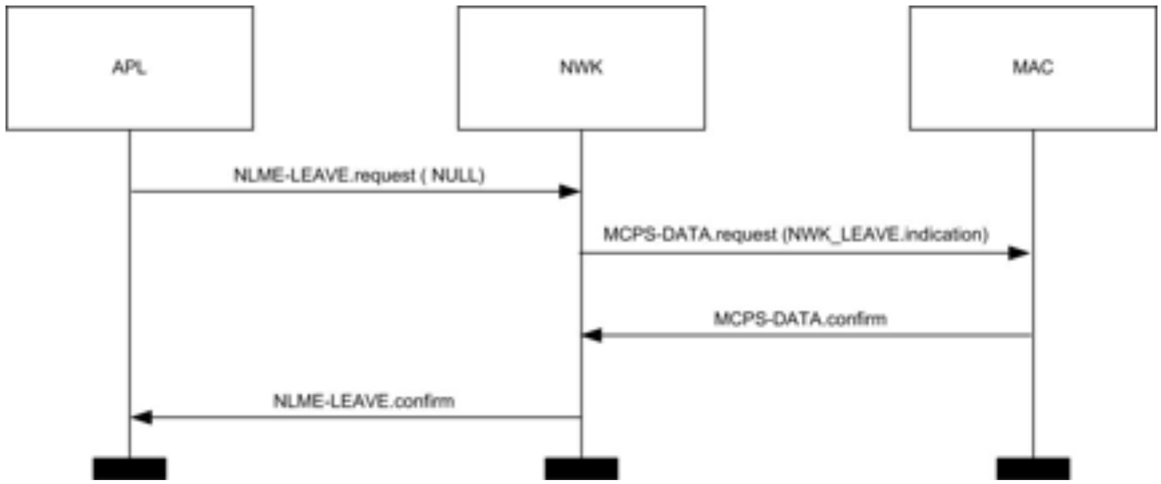


Figure 3-48. Initiation of the Leave Procedure

When the NWK layer of a Zigbee router or Zigbee coordinator, receives the NLME-LEAVE.request primitive with the DeviceAddress parameter equal to NULL or equal to the local device's IEEE address (indicating that the device is to remove itself) the device SHALL send a leave command frame using the MCPS-DATA.request primitive with the DstAddr parameter set to 0xffff indicating a MAC broadcast. The request sub-field of the command options field of the leave command frame SHALL be set to 0. The value of the remove children sub-field of the command options field of the leave command SHALL reflect the value of the RemoveChildren parameter of the NLME-LEAVE.request primitive, and the value of the Rejoin sub-field of the leave command SHALL reflect the value of the Rejoin parameter of the NLME-LEAVE.request primitive. After transmission of the leave command frame, it SHALL issue a NLME-LEAVE.confirm primitive to the higher layer with the DeviceAddress parameter equal to NULL. The Status parameter SHALL be SUCCESS if the leave command frame was transmitted successfully. Otherwise, the Status parameter of the NLME-LEAVE.confirm SHALL have the same value as the Status parameter returned by the MCPS-DATA.confirm primitive. Regardless of the Status parameter to the NLME-LEAVE.confirm, the device SHALL leave the network employing the procedure in 3.6.1.11.4.

If the device receiving the NLME-LEAVE.request primitive is a Zigbee end device, then the device SHALL send a leave command frame using the MCPS-DATA.request primitive with the DstAddr parameter set to the 16-bit network address of its parent device, indicating a MAC unicast. The request and remove children sub-fields of the command options field of the leave command frame SHALL be set to 0, and the rejoin flag in the command options SHALL be copied from the rejoin parameter of the NLME-LEAVE.request primitive. After transmission of the leave command frame, it SHALL set the *nwkExtendedPANId* attribute of the NIB to 0x0000000000000000 and issue a NLME-LEAVE.confirm primitive to the higher layer with the DeviceAddress parameter equal to NULL. The Status parameter SHALL be SUCCESS if the leave command frame was transmitted successfully. Otherwise, the Status parameter of the NLME-LEAVE.confirm SHALL have the same value as the Status parameter returned by the MCPS-DATA.confirm primitive. Regardless of the Status parameter to the NLME-LEAVE.confirm, the device SHALL leave the network employing the procedure in 3.6.1.11.4.

3.6.1.11.2 Method for a Device to Remove Its Child from the Network

This section describes how a device can initiate the removal from the network of one of its child devices in response to the receipt of an NLME-LEAVE.request primitive from the next higher layer as shown in Figure 3-49.

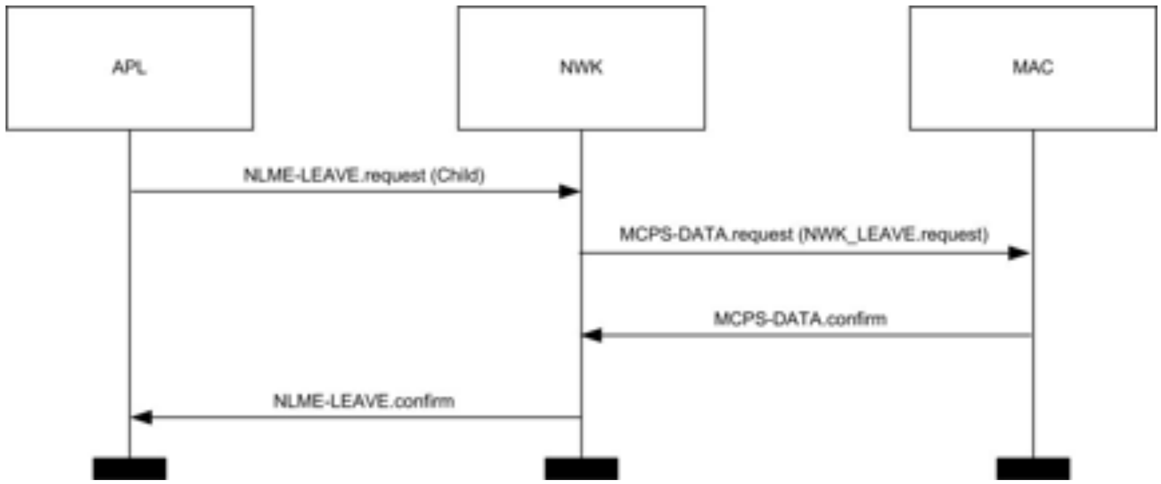


Figure 3-49. Procedure for a Device to Remove Its Child

When the NWK layer of a Zigbee coordinator or Zigbee router, receives the NLME-LEAVE.request primitive with the DeviceAddress parameter equal to the 64-bit IEEE address of a child device, if the relationship field of the neighbor table entry corresponding to that child device does not have a value of 0x05 indicating that the child has not yet authenticated, the device SHALL send a network leave command frame using the MCPS-DATA.request primitive with the DstAddr parameter set to the 16-bit network address of that child device. The request sub-field of the command options field of the leave command frame SHALL have a value of 1, indicating a request to leave the network. The value of the remove children sub-field of the command options field of the leave command SHALL reflect the value of the RemoveChildren parameter of the NLME-LEAVE.request primitive, and the value of the Rejoin sub-field of the leave command SHALL reflect the value of the Rejoin parameter of the NLME-LEAVE.request primitive.

If the relationship field of the neighbor table entry corresponding to the device being removed has a value of 0x05, indicating that it is an unauthenticated child, the device SHALL NOT send a network leave command frame.

Next, the NWK layer SHALL issue the NLME-LEAVE.confirm primitive with the DeviceAddress parameter set to the 64-bit IEEE address of the child device being removed. The Status parameter of the NLME-LEAVE.confirm primitive SHALL have a value of SUCCESS if the leave command frame was not transmitted, *i.e.* in the case of an unauthenticated child. Otherwise, the Status parameter of the NLME-LEAVE.confirm SHALL have the same value as the Status parameter returned by the MCPS-DATA.confirm primitive.

After the child device has been removed, the NWK layer of the parent SHOULD modify its neighbor table, and any other internal data structures that refer to the child device, to indicate that the device is no longer on the network. It is an error for the next higher layer to address and transmit frames to a child device after that device has been removed.

If an unauthenticated child device is removed from the network before it is authenticated, then the address formerly in use by the device being asked to leave MAY be assigned to another device that joins subsequently.

Zigbee end devices have no child devices to remove and SHOULD NOT receive NLME-LEAVE.request primitives with non-NULL DeviceAddress parameters.

3.6.1.11.3 Upon Receipt of the Leave Command Frame

Upon receipt of the leave command frame by the NWK layer via the MCPS-DATA.indication primitive, as shown in Figure 3-50, the device SHALL check the value of the request sub-field of the command options field of the command frame. If the request sub-field has a value of 0, then the NWK layer SHALL issue the NLME-LEAVE.indication primitive to the next higher layer with the device address parameter equal to the value in the source IEEE Address sub-field of the leave command frame and the rejoin parameter equal to the value in the Rejoin sub-field of the leave command frame. The device SHOULD also modify its neighbor table, to indicate that the leaving device is no longer a neighbor, regardless of the value of the rejoin flag in the primitive.

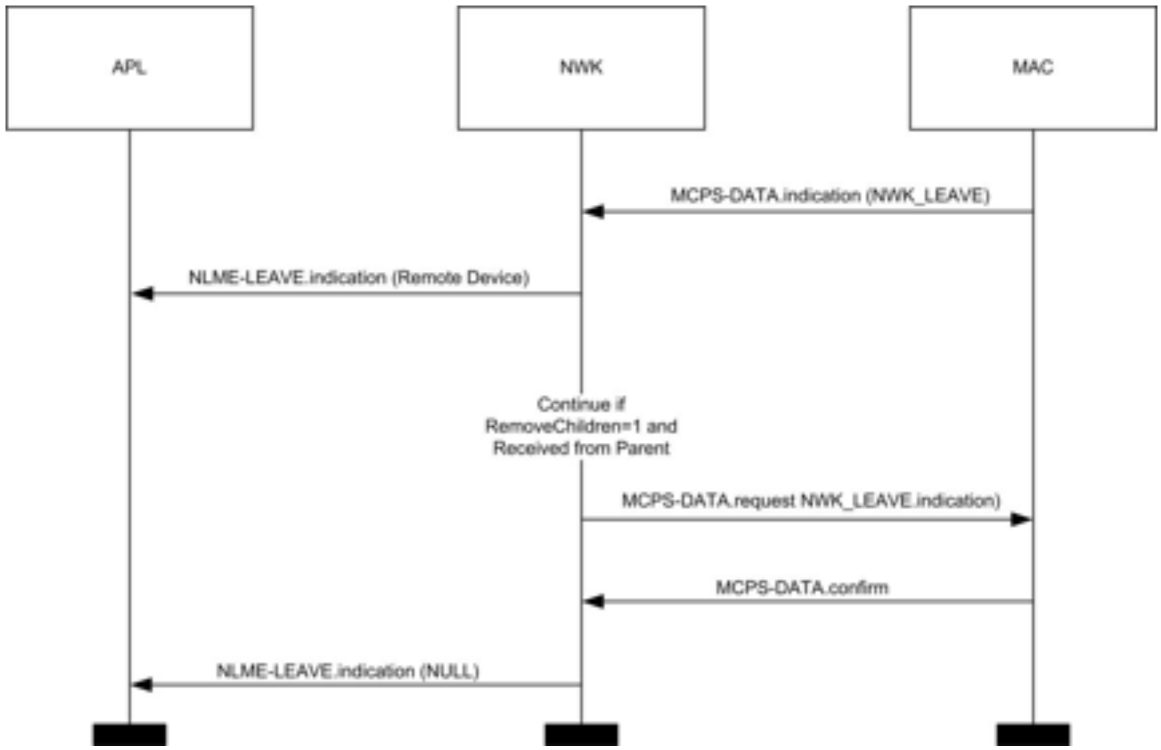


Figure 3-50. On Receipt of a Leave Command

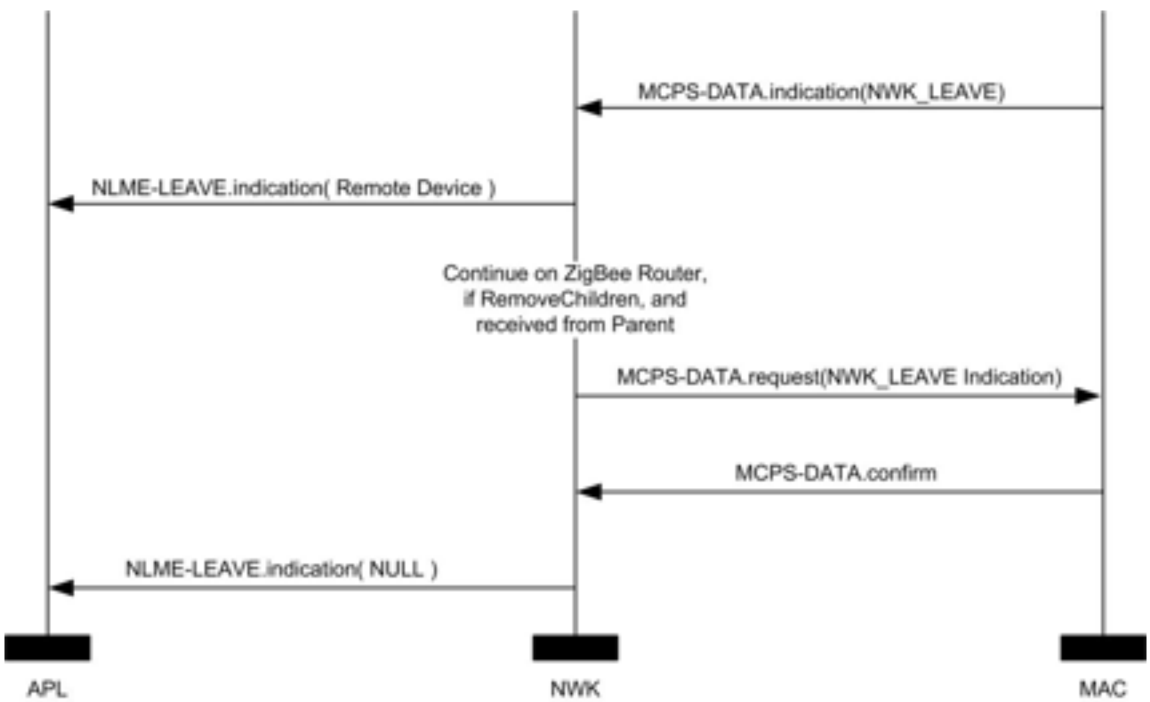


Figure 3-51. On Receipt of a Leave Command by a ZED

If, on receipt by the NWK layer of a Zigbee router of a leave command frame as described above, the SrcAddr parameter of the MCPS-DATA.indication that delivered the command frame is the 16-bit network address of the parent of the recipient, and the value of the remove children sub-field of the command options field is found to have a value of 1, then the recipient SHALL send a leave command frame using the MCPS-DATA.request primitive with the

DstAddr parameter set to 0xffff indicating a MAC broadcast. The request sub-field of the command options field of the leave command frame shall be set to 0.

The value of the remove children sub-field and the rejoin sub-field of the command options field of the outgoing leave command SHALL reflect the value of the same field for the incoming leave command frame. After transmission of the leave command frame, it SHALL set the *nwkExtendedPANId* attribute of the NIB to 0x0000000000000000 and it SHALL issue a NLME-LEAVE.indication primitive to the higher layer with DeviceAddress parameter equal to NULL.

If the request sub-field has a value of 1 then the procedure in section 3.6.1.11.3.1 shall be executed.

3.6.1.11.3.1 Validation of the Leave Request

The following procedure applies to processing of the NWK Leave (request) command frame and the ZDO Mgmt_leave_req.

1. If the device is a Zigbee Coordinator or if the message was sent to a broadcast address, the message SHALL be dropped and no further processing SHALL be done.
2. If the device is Zigbee Router, the following SHALL be performed:
    a. The device SHALL NOT consider the Relationship field within the nwkNeighborTable entry corresponding to the sending device.
    b. If the nwkLeaveRequestAllowed in the NIB is TRUE, the device SHALL perform the procedure described in 3.6.1.11.1. No further processing SHALL be done.
    c. Otherwise if nwkLeaveRequestAllowed in the NIB is FALSE, no further processing SHALL be done.
3. If the device is a Zigbee End Device, the following SHALL be performed:
    a. Examine the nwkNeighborTable for an entry where the Network Address is the same as the SrcAddr parameter of the MCPS-DATA.indication primitive that delivered the NWK command.
        i. If no entry is found, then no further processing SHALL be done.
    b. If the corresponding entry in the nwkNeighborTable has a Relationship value that is not 0x00 (neighbor is the parent), then no further processing SHALL be done.
    c. The sending device is the parent of the receiving device, the receiving device shall perform the procedure described in 3.6.1.11.1, with the following exception.: it SHOULD not send a leave command frame using the MCPS-DATA.request primitive, and otherwise continue as if it had sent the leave command frame successfully.[4] No further processing SHALL be done.
4. No further processing SHALL be done.

If a Zigbee end device receives a leave command frame as described above and the SrcAddr parameter of the MCPS-DATA.indication that delivered the command frame is the 16-bit network address of the parent of the recipient, it SHALL set the *nwkExtendedPANId* attribute of the NIB to 0x0000000000000000 and SHALL issue a NLME-LEAVE.indication primitive to the higher layer with DeviceAddress parameter equal to NULL.

The NWK layer MAY employ retry techniques, as described in section 3.2.1.1.3 to enhance the reliability of the leave procedure but, beyond this note, these mechanisms are outside the scope of this specification.

[4] CCB 3322

3.6.1.11.4 Local Process for Leaving the Network

Upon receipt of a NLME-LEAVE.request primitive or the NWK layer leave command, the following SHALL be employed.

1. If the Rejoin value is set to 1 in either the NLME-LEAVE.request primitive or the NWK Leave command, it SHALL do the following.
    a. The device MAY execute the rejoin procedure by issuing an NLME-JOIN.request with the RejoinNetwork set to 1.
    b. No further processing SHALL be done.
2. If the Rejoin value is set to 0, it SHALL clear the following values in the NIB:
    a. nwkNeighborTable
    b. nwkRouteTable
    c. nwkManagerAddr
    d. nwkUpdateId
    e. nwkNetworkAddress
    f. nwkExtendedPANID
    g. nwkRouteRecordTable
    h. nwkIsConcentrator
    i. nwkConcentratorRadius
    j. nwkSecurityMaterialSet
    k. nwkActiveKeySeqNumber
    l. nwkAddressMap
    m. nwkPANID
    n. nwkTxTotal
    o. nwkParentInformation
3. The device is no longer operating on the network.

3.6.1.12 Resetting a Device

The NWK layer of a device shall be reset immediately following initial power-up, before a join attempt to a new network and after a leave attempt where the device is not intending to rejoin the network. This process SHOULD NOT be initiated at any other time. A reset is initiated by issuing the NLME-RESET.request primitive to the NLME and the status of the attempt is communicated back via the NLME-RESET.confirm primitive. The reset process SHALL clear the routing table entries of the device.

Some devices MAY store NWK layer quantities in non-volatile memory and restore them after a reset. The WarmStart parameter of the NLME-RESET.request MAY also be used for this purpose. A device SHALL use the same address on rejoining a network and therefore SHOULD NOT discard its address on reset unless it does not intend to rejoin the same network.

3.6.1.13 Managing a PANId Conflict

Since the 16-bit PANID is not a unique number there is a possibility of a PANId conflict. The next section explains how — through the use of the Network Report and Network Update command frames — the PANId of a network can be updated.

3.6.1.13.1 Detecting a PANId Conflict

Any device that is operational on a network and receives an MLME-BEACON-NOTIFY.indication in which the PAN identifier of the beacon frame matches its own PAN identifier but the EPID value contained in the beacon payload is either not present or not equal to *nwkExtendedPANID*, SHALL be considered to have detected a PAN Identifier conflict.

The device SHALL increment the *nwkPanIdConflictsCount* NIB value. If it is already at the maximum value for the NIB value, then it SHALL NOT be incremented and stay at the maximum value.

A node that has detected a PAN identifier conflict SHALL NOT send an unsolicited Network Report Command frame of type PAN Identifier Conflict. Versions of the specification prior to Revision 23 required devices to report conflicts unsolicited to the device identified by the *nwkManagerAddr* attribute in their NIB. Therefore, it is possible that older devices will still generate these messages.

3.6.1.13.2 Upon Receipt of a Network Report Command Frame

The device identified by the 16-bit network address contained within the *nwkManagerAddr* attribute of the NIB SHALL be the recipient of network report command frames of type PAN identifier conflict.

The Network Manager SHALL notify the local application by generating an NLME-NETWORK-STATUS.indication with a status of 0x14, PAN ID Conflict Report, and the NetworkAddr equal to the device that sent the report.

The Network Manager SHOULD NOT automatically change the PAN ID of the network due to unsolicited PAN ID conflicts reported by devices in the network. PAN ID conflicts may be triggered by a malicious device that is not part of the network. A Network Manager SHOULD utilize other metrics from the higher layer to determine whether a PAN ID conflict is causing application connectivity problems. Changing a network's PAN ID SHOULD be done rarely as it will be very disruptive to network communications, especially to sleepy end devices that will not receive the notification of the PAN ID change.

If the vendor specific configurable mechanism is set to disallow automatic resolution of PAN ID conflict, the designated network layer function manager SHALL NOT unconditionally select a new 16-bit identifier for the network and SHALL NOT change to the new PAN ID immediately. The decision to change PAN IDs in this case should be based on other factors outside the scope of the stack behavior and related to the application performance.

If the designated network manager decides to resolve an actual PAN identifier conflict, it SHALL proceed as follows.[5] The new PAN identifier is chosen at random, but a check is performed to ensure that the chosen PAN identifier is not already in use in the local neighborhood and also not contained within the Report Information field of the network report command frame.

3.6.1.13.3 Changing the PAN ID of the Network

If the higher layer has determined that a PAN ID change is warranted, it MAY stage the PAN ID change by sending unicast ZDO Security_Set_Configuratiom_req to all the devices and notify the local Network Manager by setting the *nwkNextPanId* value of the NIB. Regardless of whether the next PAN ID has been staged or not, this procedure SHALL be followed:

1. If the local *nwkNextPanId* NIB value is not 0xFFFF then network manager SHALL use the value of *nwkNextPanId*, otherwise it SHALL choose a random PAN ID that is not in use.
2. Once a new PAN identifier has been selected, the designated network layer function manager SHALL first increment the NIB attribute *nwkUpdateId* (wrapping around to 0 if necessary) and then SHALL construct a network update command frame of type PAN identifier update. The update information field shall be set to the

[5] CCB 2713

value of the new PAN identifier. The network update command frame shall be sent to all devices broadcast address (0xFFFF).

3. After it sends out this command frame, the designated network layer function manager SHALL start a timer with a value equal to *nwkNetworkBroadcastDeliveryTime* OctetDurations. When the timer expires, the Zigbee coordinator SHALL change its current PAN ID to the newly selected one by reissuing the MLME-START.request with the new PANID.
4. Upon transmission of the Network Update command frame the designated network layer function manager SHALL create a NLME-NWK-STATUS.indication primitive with the NetworkAddr parameter set to 0 and the Status parameter set to PAN Identifier Update.

3.6.1.13.4 Upon Receipt of a Network Update Command Frame

On receipt of a network update command frame of type PAN identifier update the device SHALL first determine whether a PAN ID update is allowed. This provides a mechanism for the network manager to stage the PAN ID update and provide security for the update.

The receiver of the Network Update Command SHALL check the *nwkNextPanId* value of the NIB to see if it has a value of 0xFFFF or a value corresponding to the PAN Identifier Update field in the Network Update command. If neither is TRUE, the Network Update command SHALL be dropped and no further processing SHALL be done.

It SHALL then start a timer with a value equal to *nwkNetworkBroadcastDeliveryTime* OctetDurations. When the timer expires, the device SHALL change its current PAN Identifier to the value contained within the Update Information field.

Upon transmission of the network update command frame the device SHALL create a NLME-NWK-STATUS.indication primitive with the NetworkAddr parameter set to 0 and the Status parameter set to PAN Identifier Update.

Upon receipt of the Network Update command from the device identified by the *nwkManagerAddr* attribute of the NIB, the value contained in the update id field SHALL be stored in *nwkUpdateId* attribute in the NIB. The beacon payload SHALL also be updated.

3.6.1.14 Security for Changes to the PAN ID or Channel

In earlier versions of this specification a change to the PAN ID or channel change can be made with a single broadcast command. The mechanism for changing the PAN ID was to broadcast a NWK Update Command, while the mechanism for changing the channel is to broadcast a ZDO Mgmt_NWK_Update_Req with the new channel and the ScanDuration set to 0xFE.

Since these commands are broadcast there is only NWK layer security and thus any device on the network can make this change. A change to the channel or PAN ID is extremely disruptive and as such additional protections have been added in Revision 23 of this specification to allow the Trust Center, acting as the Network Manager, to restrict channel and PAN ID changes.

The effect of a single broadcast mechanism to change channels or the PAN ID is restricted by a new security mechanism that the Trust Center MAY make use of. All devices implementing Revision 23 SHALL support the new secure mechanism. Devices implementing earlier versions of the specification will only support the broadcast mechanism. The new mechanism complements the old mechanism allowing channel and PAN ID changes to work for a network of devices with mixed versions.

The mechanism for securely changing PAN IDs and channels is similar to the process of changing the network key. If the Trust Center chooses to use the mechanism it will pre-announce the change via a unicast to each device. It will do this with a ZDO Security_Set_Configuration_req containing the Next PAN ID Global TLV and or the Next Channel Change Global TLV. Devices will set their local NIB with nwkNextPanId and or nwkNextChannelChange respectively. These messages will be APS encrypted with each device specific link key.

The Trust Center SHOULD send the change to all routers in the network and MAY optionally send to end devices, but their long sleep cycles can make this too difficult to do so.

Revision 23 devices will report their successful acceptance of the next PAN ID or channel via a SUCCESS in the ZDO Security_Set_Configuration_rsp. Devices implementing early specifications will respond with a status of UN-SUPPORTED.

When the Trust Center, acting as Network Manager, wants to change the PAN ID or Channel it will broadcast the corresponding NWK Update Command or ZDO Mgmt_NWK_Update_Req. It can do this with networks of devices that implement Revision 23 or later, or devices that implement the earlier specifications.

Those devices that do not support Revision 23 or later will receive the broadcast and process it as specified in previous specifications. Those devices take the broadcast both as the notification of what channel or PAN ID to change to, and the command to switch now.

Devices supporting Revision 23 or later will:

a. If nwkNextPanId is 0xFFFF they will change their PAN ID and SHALL NOT update nwkNextPanId.

b. If nwkNextPanId is not 0xFFFF (i.e. has been reset by a Security_set_Configuration_req) and this matches the broadcast NWK Update Command the device SHALL update both the active PAN ID and the nwkNextPanId NIB value to be the new matching value.

c. Otherwise they shall ignore the broadcast NWK Update Command.

It is important to note that after the channel or PAN ID change, the nwkNextPanId and nwkNextChannelChange NIB values SHALL NOT be modified. They will remain at the same value as their current value. If value is a valid PAN ID or channel this will protect the network from further PAN ID or channel changes via broadcast messages. A Trust Center can make additional changes to the PAN ID or channel change via the same unicast mechanism as described above.

A Trust Center that wishes to make use of this functionality after forming the network SHOULD send a ZDO Security_Set_Configuration_req to each newly joined device with the current PAN and channel. This will replace the default value of 0xFFFF for nwkNextPanId and prevent any changes to the PAN or channel before the first time the Trust Center decides to make any change.

Lastly, it is always possible a device could have missed a legitimate change to the PAN ID or channel. For example, it could have been switched off when that occurred. As a result if the application initiates a rejoin via the NLME-JOIN.req primitive, the device SHALL accept a change to the PAN ID or channel upon successfully finding and rejoining the network and correctly passing security checks (e.g. an APS Transport Key encrypted with its current link key). If a device has rejoined and the channel or PAN ID has changed it does the following:

1. If the nwkNextPanId is not 0xFFFF, it SHALL be updated to match the current PAN ID after rejoining.

2. If the nwkNextChannelChange is not 0, it SHALL be updated to match the current channel after rejoining

3.6.1.15 Polling Considerations for Sleepy End Devices

During attaching RxOnWhenIdle=FALSE devices SHALL continually poll at a rate of 250 ms or less. This includes MAC Association, NWK Rejoin, NWK Commissioning, and any security commands that are being transmitted as part of authentication. The higher layer will time out the attach operation upon failure to receive the network key or failure to receive the next message in the negotiation.

3.6.2 Transmission and Reception

3.6.2.1 Transmission

Only those devices that are currently associated SHALL send data frames from the NWK layer. If a device that is not associated receives a request to transmit a frame, it SHALL discard the frame and notify the higher layer of the error by issuing an NLDE-DATA.confirm primitive with a status of INV_REQUESTTYPE.

All frames handled by or generated within the NWK layer SHALL be constructed according to the general frame format specified in Figure 3-4 and transmitted using the MAC sub-layer data service.

For data frames originating at a higher layer, the value of the source address field MAY be supplied using the Source address parameter of the NLDE-DATA.request primitive. If a value is not supplied or when the NWK layer needs to construct a new NWK layer command frame, then the source address field SHALL be set to the value of the *macShortAddress* attribute in the MAC PIB. Support of this parameter in the NLDE-DATA.request primitive is required if GP feature is to be supported by the implementation.

In addition to source address and destination address fields, all NWK layer transmissions SHALL include a radius field and a sequence number field. For data frames originating at a higher layer, the value of the radius field MAY be supplied using the Radius parameter of the NLDE-DATA.request primitive. If a value is not supplied, then the radius field of the NWK header SHALL be set to twice the value of the *nwkcMaxDepth* attribute of the NIB (see Constants and NIB Attributes). For data frames originating at a higher layer, the value of the sequence number field MAY be supplied using the Sequence number parameter of the NLDE-DATA.request primitive. If a value is not supplied or when the NWK layer needs to construct a new NWK layer command frame, then the NWK layer SHALL supply the value. Support of this parameter in the NLDE-DATA.request primitive is required if GP feature is to be supported by the implementation. The NWK layer on every device SHALL maintain a sequence number that is initialized with a random value. The sequence number SHALL be incremented by 1, each time the NWK layer supplies a new sequence number value for a NWK frame. The value of the sequence number SHALL be inserted into the sequence number field of the frame's NWK header.

Once an NPDU is complete, if security is required for the frame, it SHALL be passed to the security service provider for subsequent processing according to the specified security suite (see section 4.2.2). Security processing is not required if the SecurityEnable parameter of the NLDE-DATA.request is equal to FALSE. If the NWK security level as specified in *nwkSecurityLevel* is equal to 0, then the security sub-field of the frame control field SHALL always be set to 0.

On successful completion of the secure processing, the security suite returns the frame to the NWK layer for transmission. The processed frame will have the correct auxiliary header attached. If security processing of the frame fails and the frame was a data frame, the frame will inform the higher layer of the NLDE-DATA.confirm primitive’s status. If security processing of the frame fails and the frame is a network command frame, it is discarded and no further processing SHALL be done.

When the frame is constructed and ready for transmission, it SHALL be passed to the MAC data service. An NPDU transmission is initiated by issuing the MCPS-DATA.request primitive to the MAC sub-layer. The MCPS-DATA.confirm primitive then returns the results of the transmission.

3.6.2.2 Reception and Rejection

In order to receive data, a device SHALL enable its receiver. The next higher layer MAY initiate reception using the NLME-SYNC.request primitive. On a beacon-enabled network, receipt of this primitive by the NWK layer SHALL cause a device to synchronize with its parent’s next beacon and, optionally, to track future beacons. The NWK layer SHALL accomplish this by issuing an MLME-SYNC.request to the MAC sub-layer. On a non-beacon-enabled network, the NLME-SYNC.request SHALL cause the NWK layer of a device with *macRxOnWhenIdle* set to FALSE to poll the device’s parent using the MLME-POLL.request primitive.

On a non-beacon-enabled network, the NWK layer on a Zigbee coordinator or Zigbee router SHALL ensure, to the maximum extent feasible, that the receiver is enabled whenever the device is not transmitting. On a beacon-enabled network, the NWK layer SHOULD ensure that the receiver is enabled when the device is not transmitting during the active period of its own superframe and of its parent’s superframe. The NWK layer MAY use the *macRxOnWhenIdle* attribute of the MAC PIB for this purpose.

Once the receiver is enabled, the NWK layer will begin to receive frames via the MAC data service. On receipt of each frame, the radius field of the NWK header SHALL be decremented by 1. If, as a result of being decremented, this value falls to 0, the frame SHALL NOT, under any circumstances, be retransmitted. It MAY, however, be passed to the next higher layer or otherwise processed by the NWK layer as outlined elsewhere in this specification.

The NWK layer SHALL accept non-incremental NWK-level values in the Sequence number field of the Zigbee Network header for consecutive packets with the same value of the Source address field of the Zigbee Network header.

On receipt of a frame with the End Device Initiator sub-field of the frame control set to 1, the following processing SHALL take place.

1. If the receiving device is an end device the message SHALL be dropped and no further processing SHALL be done.
2. The receiving device SHALL search the neighbor table for an entry where the value of the Network Address matches the value of the Source Address field of the message, and the device type is 0x02 (end device).
    a. If an entry is found continue processing the frame normally.
    b. If no entry is found then the message SHALL be dropped and no further processing SHALL be done.
        i. The routing device SHALL issue either a Mgmt_Leave_req or NWK Leave command to the sender with the Rejoin parameter set to 1 and the RemoveChildren parameter set to 0. This message SHALL be sent via the MCPS-DATA.request with the IndirectTx parameter set to FALSE.
            1. Additionally, the receiving device may submit a second copy of the message to the MAC using MCPS-DATA.request with the IndirectTx set to TRUE in the case the device is a sleepy end device.[6]

The following data frames SHALL be passed to the next higher layer using the NLDE-DATA.indication primitive:

Frames with a broadcast address that matches a broadcast group of which the device is a member.

Unicast data frames and source-addressed data frames for which the destination address matches the device's network address.

If the receiving device is a Zigbee coordinator or an operating Zigbee router, that is, a router that has already invoked the NLME-START-ROUTER.request primitive, it SHALL process data frames as follows:

Messages SHALL be verified to determine if an end device has switched router parents. This is outlined in section 3.6.2.3.

Broadcast data frames SHALL be relayed according to the procedures outlined in section 3.6.5.

Unicast data frames with a destination address that does not match the device's network address SHALL be relayed according to the procedures outlined in section . (Under all other circumstances, unicast data frames SHALL be discarded immediately.)

Source-routed data frames with a destination address that does not match the device’s network address SHALL be relayed according to the procedures outlined in section 3.6.4.3.2.

The procedure for handling route request command frames is outlined in section 3.6.4.5.2.

The procedure for handling route reply command frames for which the destination address matches the device's network address is outlined in section 3.6.4.5.2.

Route reply command frames for which the destination address does not match the device's network address SHALL be discarded immediately. Network status command frames SHALL be handled in the same manner as data frames.

The NWK layer SHALL indicate the receipt of a data frame to the next higher layer using the NLDE-DATA.indication primitive.

On receipt of a frame, the NLDE SHALL check the value of the security sub-field of the frame control field. If this value is non-zero, the NLDE SHALL pass the frame to the security service provider (see section 4.2.2) for subsequent processing according to the specified security suite. If the security sub-field is set to 0, the *nwkSecurityLevel* attribute in the NIB is non-zero, the device is currently joined and authenticated, and the incoming frame is a NWK data frame, the NLDE SHALL discard the frame. If the security sub-field is set to 0, the *nwkSecurityLevel* attribute in the NIB is non-zero, and the incoming frame is a NWK command frame and the command ID is 0x06 (rejoin request), the NLDE

[6] CCB 2255

SHALL only accept the frame if it is destined to itself, that is, if it does not need to be forwarded to another device. Otherwise the frame SHALL be dropped and no further processing SHALL be done.

If the device is not joined and authenticated, or undergoing the Trust Center Rejoin process, it SHALL perform the following checks. If the frame is a NWK command where the security sub-field of the frame is set to zero then it SHALL only accept the frame if the command ID is 0x07 (rejoin response). If the frame is a NWK data frame where the security sub-field is set to 0, the device SHALL further examine the APDU and determine if it contains an APS command ID of 0x05 (Transport Key). If the message does not contain an APS Command of 0x05 (Transport Key), then the message SHALL be dropped and no further processing SHALL be done. All other messages where the security sub-field is set to 0 SHALL be dropped and no further processing SHALL be done.

3.6.2.3 Examination for End Devices that have changed Router Parents

A router and coordinator upon receipt of a NWK command or data message SHALL perform the following:

1. Search the neighbor table for an entry where the Network Address matches the value of the NWK Source field in the message. If no match is found then go to step 6.
2. Examine if the Device Type of the entry corresponds to a Zigbee End Device. If it does not, go to step 6.
3. Examine if the MAC source field of the message matches the NWK source field. If it does go to step 6.
4. If the message is a broadcast, examine if an entry exists in nwkBroadcastTransactionTable, if it does then go to step 6. If the message is a unicast, continue processing.
5. At this point this could mean the message has been relayed by another device on the network acting as the end device’s router parent; set the Relationship field of the corresponding neighbor table entry to 0x07, neighbor is a lost child..
6. Continue to process the message.

Routers and Coordinators need to allow for sufficient time (as an example 1 second) for the APS layer to process the message and examine whether the stack has an address conflict, such as is done for ZDO Device_annce. Once the APS layer has processed the message the NLME can delete all end devices from the neighbor table that changed to a different parent as indicated in the neighbor table when the Relationship field has a value of 0x07 (neighbor is a lost child). The NLME SHALL evaluate the neighbor table after the APS layer has processed the message to ascertain if the Relationship field has a value of 0x07, neighbor is a lost child. If the field is still 0x07 then the neighbor table entry SHALL be deleted. If the relationship indicates 0x08, neighbor is a child with address conflict, then an address conflict has been detected by the higher layer and SHALL follow the procedure in section 3.6.1.10.3 to resolve the address conflict.

3.6.3 Link Quality Indicator in Neighbor Table Entries

For all intents and purposes, when link quality is of relevance, e.g. for routing decisions, election of parents, etc. the network layer SHALL apply the Link Quality Assessment (LQA) metric. Filtering SHOULD be applied to raw LQA values (specified in section D.13) for links, which correspond to neighbors maintained in the neighbor table in order to mitigate the effect of singular outliers in raw LQA assessments, e.g., due to short, transitional radio interference. Such filtering is currently only recommended for neighbor table entries due to associated per-link memory requirements.

The final LQA value, based on a series of the n most recent raw LQA measurements, SHOULD be determined as follows:

$$LQA = \mathrm{med}\{LQA_{raw,1}, LQA_{raw,2}, \dots, LQA_{raw,n}\}$$

where $LQA_{raw,i}, i = 1 \dots n,$ are the n most recent raw LQA samples; and $\mathrm{med}(A)$ is the median value of set A, i.e. the “middle” value (as opposed to the average value). Only valid LQA measurements SHALL be taken into account, which might be less than n. For instance, shortly after a parent router reboots the LQA values for its child devices are undefined until it receives the first data frames from these devices. Similarly, when a new neighbor table entry is created for a sibling router, there is only one raw LQA measurement available, which is also the resulting final LQA value.

The number of the most recent samples taken into consideration SHOULD be $n = 3$, which eliminates single outliers, maintains a fast response to real changes in link quality, and keeps memory requirements to a minimum.

3.6.4 Routing

Zigbee coordinators and routers SHALL provide the following functionality:

- Relay data frames on behalf of higher layers.
- Relay data frames on behalf of other Zigbee routers.
- Participate in route discovery in order to establish routes for subsequent data frames.
- Participate in route discovery on behalf of end devices.
- Participate in route repair.
- Employ the Zigbee path cost metric as specified in route discovery.

Zigbee coordinators or routers MAY provide the following functionality:

- Maintain routing tables in order to remember best available routes.
- Initiate route discovery on behalf of higher layers.
- Initiate route discovery on behalf of other Zigbee routers.
- Initiate route repair.
- Conduct neighbor routing.

3.6.4.1 Routing Cost

The Zigbee routing algorithm uses a path cost metric for route comparison during route discovery and maintenance. In order to compute this metric, a cost, known as the link cost, is associated with each link in the path and link cost values are summed to produce the cost for the path as a whole.

More formally, if we define a path P of length L as an ordered set of devices and a link, as a sub-path of length 2, then the path cost

$$C\{P\} = \ min \begin{cases} \sum_{i=1}^{L-1} \alpha_i \cdot C\{[D_i, D_{i+1}]\} \\ 0xff \end{cases}$$

where each of the values is referred to as a link cost and α is the InterfaceLinkCostScalar for the link. The link cost for a link is mapped to the LQA value as described in Table 3-72.

Table 3-72. Link Cost to LQA Mapping

Link Cost C(i)	LQA Range
0	No LQA
1	$192 < LQA \leq 255$
2	$128 < LQA \leq 192$

Link Cost C(i)	LQA Range
3	$96 < LQA \leq 128$
4	$64 < LQA \leq 96$
5	$32 < LQA \leq 64$
6	$16 < LQA \leq 32$
7	$LQA \leq 16$

3.6.4.2 Routing Tables

A Zigbee router or Zigbee coordinator MAY maintain a routing table. The information that SHALL be stored in a Zigbee routing table entry is shown in Table 3-73. The aging and retirement of routing table entries in order to reclaim table space from entries that are no longer in use is a recommended practice; it is, however, out of scope of this specification.

Table 3-73. Routing Table Entry

Field Name	Size	Description
Destination address	2 octets	The 16-bit network address or Group ID of this route. If the destination device is a Zigbee router, Zigbee coordinator, or an end device, this field SHALL contain the actual 16-bit address of that device.
Status	3 bits	The status of the route. See Table 3-74 for values.
No route cache	1 bit	A flag indicating that the destination indicated by this address does not store source routes.
Many-to-one	1 bit	A flag indicating that the destination is a concentrator that issued a many-to-one route request.
Route record required	1 bit	A flag indicating that a route record command frame SHOULD be sent to the destination prior to the next data packet.
Expired	1-bit	When set to TRUE, this flag indicates that an expected regular many-to-one route request was missed, i.e. the last many-to-one route request for this destination was received more than *nwkConcentratorDiscoveryTime* + *nwkRouteDiscoveryTime* seconds ago. When the entry is created, this field is initially set to FALSE. This flag only has meaning for entries, which have the many-to-one field set to TRUE
Sequence Number Valid	1 bit	A flag indicating that the Sequence Number is valid.

Field Name	Size	Description
Next-hop address	2 octets	The 16-bit network address of the next hop on the way to the destination.
Sequence Number	2 octets	The 16-bit sequence number associated with this entry, obtained from the last route message that successfully updated this entry and conveyed a sequence number. Notice that routers prior to R23 did neither maintain nor convey a sequence number. The value stored in this field is only valid if the Sequence Number Valid flag is set.
TotalUsageCount	4 octets	A 32-bit saturating counter, which is incremented whenever this routing table entry is used to forward a data packet towards its destination.
RecentActivity	1 octet	An 8-bit saturating counter, which is pre-loaded with nwkRouterAge-Limit when the routing table entry is created; incremented whenever this routing table entry is used to forward a data packet towards its destination; and decremented unconditionally once every nwkLinkStatusPeriod. A value of 0 indicates no packets have recently been forwarded along this route.

Table 3-74 enumerates the values for the route status field.

Table 3-74. Route Status Values

Numeric Value	Status
0x0	ACTIVE
0x1	DISCOVERY_UNDERWAY
0x2	DISCOVERY_FAILED
0x3	INACTIVE
0x4 – 0x7	Reserved

This section describes the routing algorithm. The term “routing table capacity” is used to describe a situation in which a device has the ability to use its routing table to establish a route to a particular destination device. A device is said to have routing table capacity if:

- It is a Zigbee coordinator or Zigbee router.
- It maintains a routing table.
- It has a free routing table entry or it already has a routing table entry corresponding to the destination.

If a Zigbee router or Zigbee coordinator maintains a routing table, it SHALL also maintain a route discovery table containing the information shown in Table 3-75. Routing table entries are long-lived, while route discovery table entries last only as long as the duration of a single route discovery operation and MAY be reused.

Table 3-75. Route Discovery Table Entry

Field Name	Size (octets)	Description
Route request ID	1	A sequence number for a route request command frame that is incremented each time a device initiates a route request. Notice that this 8-bit identifier is distinct from the 16-bit Routing Sequence Number. The former is used to discern route requests originating in a particular router; the latter is used to identify stale routing information.
Source address	2	The 16-bit network address of the route request's initiator.
Sender address	2	The 16-bit network address of the device that has sent the most recent lowest cost route request command frame corresponding to this entry's route request identifier and source address. This field is used to determine the path that an eventual route reply command frame SHOULD follow.
Forward cost	1	The accumulated path cost from the source of the route request to the current device.
Residual cost	1	The accumulated path cost from the current device to the destination device.
Expiration time	2	A countdown timer indicating the number of milliseconds until route discovery expires. The initial value is *nwkcRouteDiscoveryTime.*

A device is said to have "route discovery table capacity" if:

- It has a free entry in its route discovery table.
- Routing table capacity and route discovery table capacity are separate resources of the device.

During route discovery, the information that a Zigbee router or Zigbee coordinator is required to maintain in order participate in the discovery of a particular route is distributed between a routing table entry and a route discovery table entry. Once discovery has been completed, only the routing table entry need be maintained in order for the NWK layer to perform routing along the discovered route. Throughout this section, references are made to this relationship between a routing table entry and its "corresponding" route discovery table entry and vice versa. The maintenance of this correspondence is up to the implementer since entries in the tables have no elements in common, but it is worth noting in this regard that the unique "keys" that define a route discovery are the source address of the route discovery command frame and the route request ID generated by that device and carried in the command frame payload.

If a device has the capability to initiate a many-to-one route request, it MAY also maintain a route record table (see Table 3-63).

3.6.4.3 Upon Receipt of a Unicast Frame

On receipt of a unicast frame from the MAC sub-layer, or an NLDE-DATA.request from the next higher layer, the NWK layer routes it according to the following procedure.

If the receiving device is a Zigbee router or Zigbee coordinator, and the destination of the frame is a Zigbee end device and also the child of the receiving device, the frame SHALL be routed directly to the destination using the MCPS-DATA.request primitive, as described in section . The frame SHALL also set the next hop destination address equal to the final destination address. Otherwise, for purposes of the ensuing discussion, define the *routing address* of a device to be its network address if it is a router or the coordinator or an end device. Define the *routing destination* of a frame to be the routing address of the frame's NWK destination.

A Zigbee router or Zigbee coordinator MAY check the neighbor table for an entry corresponding to the routing destination of the frame. If there is such an entry, the device MAY route the frame directly to the destination using the MCPS-DATA.request primitive as described in section 3.6.1.6.1.1.

A device that has routing capacity SHALL check its routing table for an entry corresponding to the routing destination of the frame. If there is such an entry, and if the value of the route status field for that entry is ACTIVE, the device SHALL relay the frame using the MCPS-DATA.request primitiveIt SHALL increment the routing table entry's TotalUsageCount and RecentActivity fields, saturating at the maximum permissible value, and also increment the next hop's RouterOutboundActivity counter, again saturating at the maximum permissible value. If the many-to-one field of the routing table entry is set to TRUE, the NWK SHALL follow the procedure outlined in section 3.6.4.5.5 to determine whether a route record command frame SHALL be sent.

When relaying a unicast frame, the SrcAddrMode and DstAddrMode parameters of the MCPS-DATA.request primitive SHALL both have a value of 0x02, indicating 16-bit addressing. The SrcPANId and DstPANId parameters SHALL both have the value provided by the macPANId attribute of the MAC PIB for the relaying device. The SrcAddr parameter SHALL be set to the value of *macShortAddress* from the MAC PIB of the relaying device, and the DstAddr parameter SHALL be the value provided by the next-hop address field of the routing table entry corresponding to the routing destination. Bit *b0* of the TxOptions parameter SHOULD be set to 1, indicating acknowledged transmission.

The NWK Sequence Number of a replayed packet SHALL NOT be changed by a router device relaying the packet. The router device relaying a packet SHALL leave the NWK Sequence Number of the originating device in the NWK Sequence Number field.

If the device has a routing table entry corresponding to the routing destination of the frame but the value of the route status field for that entry is DISCOVERY_UNDERWAY, the device SHALL determine if it initiated the discovery by consulting its discovery table. If the device initiated the discovery, the frame SHALL be treated as though route discovery has been initiated for this frame, otherwise, the device SHALL initiate route discovery as described in section 3.6.4.5.1. The frame MAY optionally be buffered pending route discovery

If the device does not have a routing table entry for the routing destination with a status value of ACTIVE, and it received the frame from the next higher layer, it SHALL check its source route table for an entry corresponding to the routing destination. If such an entry is found and the length is less than *nwkMaxSourceRoute*, the device SHALL transmit the frame using source routing as described in section 3.6.4.3.1. If the device does not have a routing table entry for the routing destination and it is not originating the frame using source routing, it SHALL examine the discover route sub-field of the NWK header frame control field. If the discover route sub-field has a value of 0x01, the device SHALL initiate route discovery, as described in section 3.6.4.5.1. If the discover route sub-field has a value of 0 and there is no routing table corresponding to the routing destination of the frame, the frame SHALL be discarded and the NLDE SHALL issue the NLDE-DATA.confirm primitive with a status value of ROUTE_ERROR.

A device without routing capacity SHALL discard the frame. If the frame is the result of an NLDE-DATA.request from the NHL of the current device, the NLDE SHALL issue the NLDE-DATA.confirm primitive with a status value of ROUTE_ERROR. If the frame is being relayed on behalf of another device, the NLME SHALL issue a network status command frame destined for the device that is the source of the frame with a status of 0x04, indicating a lack of routing capacity. It SHALL also issue the NLME-NWK-STATUS.indication to the next higher layer with the NetworkAddr parameter equal to the 16-bit network address of the frame, and the Status parameter equal to 0x04, indicating a lack of routing capacity.

If the destination is an end device, delivery of the frame can fail due to the *macRxOnWhenIdle* state of the child device.

If the NWK layer on a Zigbee router or Zigbee coordinator fails to deliver a unicast frame for any reason, the router or coordinator SHALL make additional attempts by calling MCPS-DATA.request up to *nwkcUnicastRetries*. Each attempt SHALL be delayed by at least *nwkcUnicastRetryDelay* and SHALL be re-encrypted with the newest network frame counter. After all NWK Retries have been exhausted the device will make its best effort to report the failure. No failure SHOULD be reported as the result of a failure to deliver a NLME-NWK-STATUS. The failure reporting MAY take one of two forms. If the failed frame was being relayed as a result of a request from the next higher layer, then the NWK layer SHALL issue an NLDE-DATA.confirm with the error to the next higher layer. The value of the NetworkAddr parameter of the primitive SHALL be the intended destination of the frame. If the frame was being relayed on behalf of another device, then the relaying device SHALL send a network status command frame back to the source of the frame. The destination address field of the network status command frame SHALL be taken from the destination address field of the failed data frame.

In either case, the reasons for failure that MAY be reported appear in Table 3-52.

If SrcAddrMode and DstAddrMode are both equal to 0x02, i.e. the frame is a unicast addressed frame, the NWK layer SHALL increment the RouterInboundActivity of the neighbor table entry that belongs to the SrcAddr parameter of the MCPS-DATA.indication, if such an entry exists.

3.6.4.3.1 Originating a Source Routed Data Frame

If, on receipt of a data frame from the next higher layer, it is determined that the frame SHOULD be transmitted using source routing as described above, the source route SHALL be retrieved from the route record table.

If there are no intermediate relay nodes, the frame SHALL be transmitted directly to the routing destination without source routing by using the MCPS-DATA.request primitive, with the DstAddr parameter value indicating the routing destination.

If there is at least one relay node, the source route flag of the NWK header frame control field SHALL be set, and the NWK header source route subframe SHALL be present. The relay count sub-field of the source route subframe SHALL have a value equal to the number of relays in the relay list. The relay index sub-field SHALL have a value equal to 1 less than the number of relays. The relay list sub-field SHALL contain the list of relay addresses, least significant octet first. The relay closest to the destination SHALL be listed first. The relay closest to the originator SHALL be listed last.

The device SHALL relay the frame using the MCPS-DATA.request primitive. The DstAddr parameter SHALL have the value of the final relay address in the relay list.

3.6.4.3.2 Relaying a Source Routed Data Frame

Upon receipt of a source routed data frame from the MAC sub-layer as described in section , if the relay index sub-field of the source route sub-frame has a value of 0, the device SHALL check the destination address field of the NWK header of the frame. If the destination address field of the NWK header of the frame is equal in value to the *nwkNetworkAddress* attribute of the NIB, then the frame SHALL be passed to the next higher layer using the NLDE-DATA.indication primitive. If the destination address field is not equal to the *nwkNetworkAddress* attribute of the NIB, and the receiving device is a Zigbee router or Zigbee coordinator, the device SHALL relay the frame directly to the NWK header destination using the MCPS-DATA.request primitive, otherwise the frame SHALL be discarded silently.

If the relay index sub-field has a value other than 0, the device SHALL compare its network address with the address found at the relay index in the relay list. If the addresses do not match, the frame SHALL be discarded and no further action SHALL be taken. Otherwise, as long as the destination address is not the address of an end device where the relaying device is the parent, the device SHALL decrement the relay index sub-field by 1, and relay the frame to the address immediately prior to its own address in the relay list sub-field. If the destination address of the frame is an end device child of the relaying device, the frame SHALL be unicast using the MCPS-DATA.request primitive.

When relaying a source routed data frame, the NWK layer of a device SHALL also examine the routing table entry corresponding to the source address of the frame. If the no route cache field of the routing table entry has a value of FALSE, then the route record required field of the routing table entry SHALL be set to FALSE.

3.6.4.4 Link Status Messages

Wireless links can be asymmetric, that is, they can work well in one direction but not the other. This can cause route replies to fail, since they travel backwards along the links discovered by the route request.

For many-to-one routing and two-way route discovery, it is a requirement to discover routes that are reliable in both directions. To accomplish this, routers exchange link cost measurements with their neighbors by periodically transmitting link status frames as a one-hop broadcast. The reverse link cost information is then used during route discovery to ensure that discovered routes use high-quality links in both directions.

3.6.4.4.1 Initiation of a Link Status Command Frame

When joined to a network, a Zigbee router or coordinator SHALL periodically send a link status command every *nwkLinkStatusPeriod* seconds, as a one-hop broadcast without retries. It MAY be sent more frequently if desired. Random jitter SHOULD be added to avoid synchronization with other nodes. See section 3.4.8 for the link status command frame format.

End devices do not send link status command frames.

3.6.4.4.2 Upon Receipt of a Link Status Command Frame

Upon receipt of a link status command frame by a Zigbee router or coordinator, the age field of the neighbor table entry, if any, corresponding to the transmitting device and corresponding interface is reset to 0. If the link status message is marked as the first fragment:

1. The RouterAge field of this neighbor table entry is incremented, unless it had already reached the maximum permissible value for this field before, and
2. The RouterConnectivity field is determined by subtracting the maximum of the incoming and outgoing cost fields of each entry in the link status list, which exhibits a non-zero outgoing cost, from 7 and accumulating the difference, this is

$$RouterConnectivity = \sum_{\forall i|\ C_i^{out} \neq 0} \left\{7 - \max\left(C_i^{in}, C_i^{out}\right)\right\}$$

   where C_i^{in} is the incoming cost of the *i*-th element in the link status list, and C_i^{out} is the outgoing cost of the *i*-th element in the link status list, and
3. The RouterNeighborSetDiversity field is determined by regarding the set of neighbors advertised in the link status, which are also not neighbors of the local device, that is

$$D = A \setminus B|_{C_{a\in A}^{out} \neq 0, C_{b\in B}^{out} \neq 0}$$

   where A is the set of addresses in the link status list with non-zero outgoing cost and B is the set of neighbors in the local neighbor table with non-zero outgoing cost, and subtracting the maximum of the incoming and outgoing cost fields of each element of this set from 7 and accumulating the difference, that is

$$RouterNeighborSetDiversity = \sum_{\forall d \in D} \left\{7 - \max\left(C_d^{in}, C_d^{out}\right)\right\}$$

If no such entry in the neighbor table existed, one SHALL be created with the RouterAge field initialized to 0, and RouterConnectivity and RouterNeighborSetDiversity calculated as above. The list of addresses covered by a frame is determined from the first and last addresses in the link status list, and the first frame and last frame bits of the command options field. If the receiver's network address is outside the range covered by the frame, the frame is discarded and processing is terminated, unless the link status list was completely empty. If the receiver's network address falls within the range covered by the frame, then the link status list is searched. If the receiver's address is found, the outgoing cost field of the neighbor table entry corresponding to the sender is set to the incoming cost value of the link status entry. If the receiver's address is not found, the outgoing cost field is set to 0. Whenever the outgoing cost field is set to 0, the link to this neighbor SHOULD be considered unidirectional or completely broken, and, as a result, all routing table entries where this neighbor appears as a next hop MAY be considered invalid and their statuses MAY be changed to INACTIVE.

If the link status field was empty, and a neighbor table entry corresponding to the transmitting device did not exist, the receiver SHALL initiate a gratuitous Link Status command frame at a point in time randomly chosen within nwkcMinRouterBootstrapJitter and nwkcMaxRouterBootstrapJitter.

End devices do not process link status command frames.

3.6.4.4.3 Maintaining Neighbor Table Entries

The set of current neighbors maintained at each router determines network connectivity and impacts overall network performance. Care SHALL be taken to select the optimal set of neighbors once the neighbor table has reached its implementation-specific capacity. All routers and the coordinator SHALL converge on a stable mesh backbone for the network using the distributed algorithm described here, which elevates a number of sibling routers to backbone mesh siblings.

A scoring heuristic is applied to rank neighbors by their relevance for optimal network performance on a global scale – despite only having local network topology information available in the neighbor table. In order to assess the relevance of each candidate neighbor, a portion of the neighbor table is reserved for accepting new neighbors, for instance to allow exchange of link status command frames during an initial observation window. Below the implementation-specific size limit of the neighbor table, all candidate neighbors can be accommodated and SHOULD therefore be accepted and there is no requirement to reject new neighbors at this stage.

3.6.4.4.3.1 Ranking Candidate Devices in Radio Range for Addition to or Removal from the Neighbor Table

A coordinator or router that receives a link status message SHALL create or maintain a corresponding entry in the neighbor table as specified in section 3.6.1.7 if it has sufficient space available; otherwise, if its neighbor table is already at full capacity, it SHALL determine whether it MAY safely remove one of the entries in the neighbor table without disturbing network performance in order to make room for the new neighbor. It MAY remove only those neighbors, which are (i) none of the distinguished backbone mesh routers, (ii) none of the end-device child devices, and (iii) have been observed for a certain period of time.

In case the local router is operating at capacity, it SHALL first determine the set of peer router devices currently in its assessment pool by including all neighbors with a relationship of 'sibling', where the TotalRouterAgeField is higher than *nwkcMinRouterObservationTime*. If this assessment pool is empty, i.e. all neighbors under observation have just recently been added, the link status message SHALL be dropped and no further processing SHALL be done.

The local device then ranks the candidate sibling routers in its assessment pool by regarding RouterNeighborSetDiversity, RouterConnectivity, RouterTotalAge, RouterInboundActivity, RouterOutboundActivity, LQA, TransmitFailure and the TotalUsageCount and RecentActivity fields of any routing table entries on record where the sibling router is listed as the next hop.

$$rank(n_i) = 4 * RouterNeighborSetDiversity_i + RouterConnectivity_i + 8 * RouterOutboundActivity_i + 8 * RouterInboundActivity_i$$

The lowest ranking neighbor SHALL then be removed from the neighbor table, and a new neighbor table entry SHALL be created for the candidate router according to section 3.6.4.4.2.

Once the TotalRouterAge of an entry for a sibling router exceeds *nwkcBackboneMeshFormationTime* it SHALL be considered for promotion to a backbone mesh sibling router applying the same ranking paradigm as above, now for the combined set of mesh backbone routers and routers in the assessment pool. The relationship field of the entries in the neighbor table, which correspond to the topmost *nwkcBackboneMeshRouterNeighbors* entries of the resulting set SHALL be set to 'backbone mesh sibling', whereas the remaining elements SHALL be set to 'sibling'. This classification MAY be revised during normal operation when any of the metrics, which influence the ranking, change considerably.

3.6.4.4.4 Aging the Neighbor Table

For devices using link status messages, the age fields for routers in the neighbor table are incremented every *nwkLinkStatusPeriod*. If the value exceeds *nwkRouterAgeLimit*, the outgoing cost field of the neighbor table entry is set to 0. In other words, if a device fails to receive *nwkRouterAgeLimit* link status messages from a router neighbor in a row, the old outgoing cost information is discarded. In this case, the neighbor entry is considered stale and MAY be reused if necessary to make room for a new neighbor. End devices do not issue link status messages and therefore SHOULD never be considered stale.

3.6.4.5 Route Discovery and Advertisement

Route discovery is the procedure whereby network devices cooperate to find and establish routes through the NWK. *Unicast route discovery* is always performed with regard to a particular source device and a particular destination device. *Many-to-one route advertisement, also known as many-to-one route discovery,* is performed by a source device to establish routes to itself from all Zigbee routers and Zigbee coordinator, within a given radius. A source device that initiates a many-to-one route advertisement is designated as a concentrator and referred to as such in this document. Throughout section 3.6.4.5 *a destination address* MAY be a 16-bit broadcast address or the 16-bit network address of a particular device. A route request command whose destination address is the routing address of a particular device is a *unicast route request*. A route request command whose destination address sub-field is a broadcast address (see Table 3-76) is a *many-to-one route request*.

Note that on RREP new frames SHALL be created at every hop. In all other cases the packets SHALL NOT be not considered a "new" frame. A new frame SHALL be one with a new route request identifier. For RREP the sequence number is regenerated every hop. For RREC the sequence number does not change with every hop.

3.6.4.5.1 Initiation of Route Discovery

There are 3 cases when route discovery is initiated.

1. An NLME-ROUTE-DISCOVERY.request primitive is received from the next higher layer AND DstAddrMode is set to 0x02.
2. An NLDE-DATA.request is received from the next higher layer AND all of the following are TRUE
   a. The DstAddrMode parameter is set to 0x02 (16-bit network address)
   b. The DiscoverRoute parameter set to 0x01
   c. There is no associated routing table entry for the DstAddr parameter.
3. Upon receipt of a frame from the MAC layer where ALL of the following are TRUE.
   a. The discover route sub-field in the NWK header is set to 0x01
   b. The value of the NWK source address of the NWK Header of the received frame is the same as a 16-bit network address of an end device child (i.e. an entry in the *nwkNeighborTable* where Device Type is to 0x2, Zigbee End Device).

In all other circumstances a route discovery SHALL NOT be initiated.

If the device initiating route discovery is currently operating as a concentrator, as indicated by the *nwkIsConcentrator* flag, and has not been specifically instructed by the NHLE to seek a normal ad-hoc route versus a source route, it SHOULD prefer discovery of source routes over discovery of ad-hoc routes. It still MAY perform normal ad-hoc route discovery, e.g. to avoid the per-frame source route overhead.

If the device initiating route discovery has no routing table entry corresponding to the routing address of the destination device, and intends to perform a normal ad-hoc route discovery, it SHALL establish a routing table entry with status equal to DISCOVERY_UNDERWAY. If the device has an existing routing table entry corresponding to the routing address of the destination with status equal to ACTIVE, that entry SHALL be used and the status field of that entry SHALL retain its current value. If it has an existing routing table entry with a status value other than ACTIVE, that entry SHALL be used and the status of that entry SHALL be set to DISCOVERY_UNDERWAY. The device SHALL also establish the corresponding route discovery table entry if one with the same initiator and route request ID does not already exist.

Each device issuing a route request command frame SHALL maintain a counter used to generate route request identifiers. When a new route request command frame is created, the route request counter is incremented and the value is stored in the device's route discovery table in the Route request identifier field. The device SHALL increment *nwkRoutingSequenceNumber*. Other fields in the routing table and route discovery table are set as described in section 3.6.4.2.

The NWK layer MAY choose to buffer the received frame pending route discovery.

Once the device creates the route discovery table and routing table entries, the route request command frame SHALL be created with the payload depicted in . The individual fields are populated as follows:

- The command frame identifier field SHALL be set to indicate the command frame is a route request, see Table 3-50.
- The Route request identifier field SHALL be set to the value stored in the route discovery table entry.

- The destination address field SHALL be set in accordance with the destination address for which the route is to be discovered.
- The path cost field SHALL be set to 0.

Once created, the route request command frame is ready for broadcast and is passed to the MAC sub-layer using the MCPS-DATA.request primitive.

When broadcasting a route request command frame at the initiation of route discovery, the NWK layer SHALL retry the broadcast *nwkcInitialRREQRetries* times after the initial broadcast, resulting in a maximum of *nwkcInitialRREQRetries* + 1 transmissions. The retries will be separated by a time interval of *nwkcRREQRetryInterval* OctetDurations.

The many-to-one route advertisement procedure SHALL be initiated by the NWK layer of a Zigbee router or coordinator on receipt of an NLME-ROUTE-DISCOVERY.request primitive from the next higher layer where the DstAddrMode parameter has a value of 0x00. A many-to-one route request command frame is not retried; however, a discovery table entry is still created to provide loop detection during the *nwkcRouteDiscoveryTime* period. If the NoRouteCache parameter of the NLME-ROUTE-DISCOVERY.request primitive is TRUE, the many-to-one sub-field of the command options field of the command frame payload SHALL be set to 2. Otherwise, the many-to-one sub-field SHALL be set to 1. Note that in this case, the NWK layer should maintain a route record table. The destination address field of the NWK header SHALL be set to 0xfffc, the all-router broadcast address. The broadcast radius SHALL be set to the value in *nwkConcentratorRadius*. A source device that initiates a many-to-one route advertisement is designated as a concentrator and referred to as such in this document and the NIB attribute *nwkIsConcentrator* should be set to TRUE. If a device has *nwkIsConcentrator* equal to TRUE and there is a non-zero value in *nwkConcentratorDiscoveryTime*, the network layer should issue a route request command frame each *nwkConcentratorDiscoveryTime*, making sure that any two consecutive many-to-one route request commands with different route request identifier are separated in time by at least *nwkConcentratorDiscoverySeparation*.

Upon receipt of a route request command frame, if the device is an end device, it SHALL drop the frame. Otherwise, it SHALL determine if it has routing capacity.

If the device does not have routing capacity and the route request is a many-to-one-route request, the route request SHALL be discarded and route request processing SHALL be terminated.

3.6.4.5.1.1 Initiating a Route Reply or Reactive Many-to-One Route Request

The device SHALL check if it is the intended destination. If it is the intended destination and the device is currently operating as a concentrator, it SHALL check if the destination of the command frame is one of its end device children by comparing the destination address field of the route request command frame payload with the address of each of its end device children, if any. If either the device or one of its end device children is the destination of the route request command frame, and it is not issuing a reactive many-to-one route request, it SHALL reply with a route reply command frame.

When replying to a route request with a route reply command frame, the device SHALL construct a frame with the frame type field set to 0x01. The route reply's source address SHALL be set to the 16-bit network address of the device creating the route reply and the destination address SHALL be set to the calculated next hop address, considering the originator of the route request as the destination. The link cost from the next hop device to the current device shall be computed as described in section 3.6.4.1 and inserted into the path cost field of the route reply command frame. The device SHALL increment *nwkRoutingSequenceNumber*.The route reply command frame SHALL be unicast to the next hop device by issuing an MCPS-DATA.request primitive.

3.6.4.5.1.2 Routing and Route Discovery Table Maintenance, Route Request Forwarding

If the device is not the destination of the route request command frame, the device SHALL compute the link cost from the previous device that transmitted the frame, as described in section 3.6.4.1. This value SHALL be added to the path cost value stored in the route request command frame.

If the device does have routing capacity and the received request is a unicast route request, the device SHALL check if it is the destination of the command frame by comparing the destination address field of the route request command frame payload with its own address. It SHALL also check if the destination of the command frame is one of its end device children by comparing the destination address field of the route request command frame payload with the address of each of its end device children, if any. If neither the device nor one of its end device children is the destination of the route request command frame, the device SHALL determine if a route discovery table (see Table 3-75) entry exists with the same route request identifier and source address field. If no such entry exists, one SHALL be created.

For both many-to-one and regular router requests, upon receipt of a route request command frame, the neighbor table is searched for an entry corresponding to the transmitting device. If no such entry is found, or if the outgoing cost field of the entry has a value of 0, the frame is discarded and route request processing is terminated. The maximum of the incoming and outgoing costs for the neighbor is used for the purposes of the path cost calculation, instead of the incoming cost. This includes the value used to increment the path cost field of the route request frame prior to retransmission.

When creating the route discovery table entry, the fields are set to the corresponding values in the route request command frame. The only exception is the forward cost field, which is determined by using the previous sender of the command frame to compute the link cost, as described in section 3.6.4.1, and adding it to the path cost contained the route request command frame. The result of the above calculation is stored in the forward cost field of the newly created route discovery table entry. The device SHALL also create a routing table entry with the destination address field set to the source address of the route request command frame and the next hop field set to the address of the previous device that transmitted the command frame. The status field SHALL be set to ACTIVE. The device SHALL then issue a route reply command frame to the source of the route request command frame. In the case that the device already has a route discovery table entry for the source address and route request identifier pair, the device SHALL determine if the path cost in the route request command frame is less than the forward cost stored in the route discovery table entry. The comparison is made by first computing the link cost from the previous device that sent this frame, as described in section 3.6.4.1, then adding it to the path cost value in the route request command frame. If this value is greater than the value in the route discovery table entry, the frame SHALL be dropped and no further processing SHALL be done. Similarly, if incoming route information is considered unsuitable as defined in section 3.6.4.5.3, the frame SHALL be dropped and no further processing SHALL be done. Otherwise, the forward cost and sender address fields in the route discovery table are updated with the new cost and the previous device address from the route request command frame.

If the received route request command frame is a unicast route request, the device SHALL also create a routing table entry with the destination address field set to the source address of the route request command frame and the next hop field set to the address of the previous device that transmitted the command frame. The status field SHALL be set to ACTIVE. The device SHALL then respond with a route reply command frame. In either of these cases, if the device is responding on behalf of one of its end device children, the responder address in the route reply command frame payload SHALL be set equal to the address of the end device child and not of the responding device.

When a device with routing capacity is not the destination of the received route request command frame, it SHALL determine if a route discovery table entry (see Table 3-75) exists with the same route request identifier and source address field. If no such entry exists, one SHALL be created. The route request timer SHALL be set to expire in *nwkcRouteDiscoveryTime* OctetDurations. If a routing table entry corresponding to the routing address of the destination exists and its status is not ACTIVE, the status SHALL be set to DISCOVERY_UNDERWAY. If no such entry exists and the frame is a unicast route request, an entry SHALL be created and its status set to DISCOVERY_UNDERWAY. If the frame is a many-to-one route request, the device SHALL also create a routing table entry with the destination address field equal to the source address of the route request command frame by setting the next hop field to the address of the previous device that transmitted the command frame. If the frame is a many-to-one route request (*i.e.* the many-to-one sub-field of the command options field of the command frame payload has a non-zero value), the many-to-one field in the routing table entry SHALL be set to TRUE, the route record required field SHALL be set to TRUE, and the no route cache flag SHALL be set to TRUE if the many-to-one sub-field of the command options field of the command frame payload has a value of 2 or to FALSE if it has a value of 1. If the routing table entry is new, or if the no route cache flag is set to TRUE, or if the next hop field changed, the route record required field SHALL be set to TRUE.. The status field SHALL be set to ACTIVE.

If an entry in the route discovery table already exists, the path cost in the route request command frame shall be compared to the forward cost value in the route discovery table entry. The comparison is made by computing the link cost from the previous device, as described in section 3.6.4.1, and adding it to the path cost value in the route request command frame. If this path cost is greater, the route request command frame is dropped and no further processing SHALL be done. Similarly, if incoming route information is considered unsuitable as defined in section 3.6.4.5.3, the frame SHALL be dropped and no further processing SHALL be done. Otherwise, the forward cost and sender address fields in the route discovery table are updated with the new cost and the previous device address from the route request command frame. Additionally, the path cost field in the route request command frame SHALL be updated with the cost computed for comparison purposes. If the received route request command frame is a unicast route request, the device SHALL also update any routing table entry with the destination address field set to the source address of the route request command frame, and the next hop field set to the address of the previous device that transmitted the command frame. The status field SHALL be set to ACTIVE.

3.6.4.5.1.3 Processing Reactive Many-To-One Route Requests

If the frame is a many-to-one route request (i.e. the many-to-one sub-field of the command options field of the command frame payload has a non-zero value), and the receiving device has changed the status field of the corresponding routing table entry from DISCOVERY_UNDERWAY to ACTIVE due to the processing steps above, the receiving device SHALL process any NLDE data requests that might be pending. It SHALL also issue an NLME-ROUTE-DISCOVERY.confirm primitive with a status of SUCCESS for each related route discovery table entry, if no such confirmation has been issued before. It SHALL keep these route discovery table entries until they expire.

3.6.4.5.1.4 Transmitting Relayed Route Requests

The device SHALL then broadcast the route request command frame using the MCPS-DATA.request primitive.

When broadcasting a route request command frame, the NWK layer SHALL delay retransmission by a random jitter amount calculated using the formula:

$$2 \times R[nwkcMinRREQJitter, nwkcMaxRREQJitter]$$

where R is a random function on the interval. The units of this jitter amount are milliseconds. Implementers MAY adjust the jitter amount so that route request command frames arriving with large path cost are delayed more than frames arriving with lower path cost. The NWK layer SHALL retry the broadcast *nwkcRREQRetries* times after the original relay resulting in a maximum of *nwkcRREQRetries* + 1 relays per relay attempt. Implementers MAY choose to discard route request command frames awaiting retransmission in the case that a frame with the same source and route request identifier arrives with a lower path cost than the one awaiting retransmission.

The device SHALL also set the status field of the routing table entry corresponding to the routing address of the destination field in the payload to DISCOVERY_UNDERWAY. If no such entry exists, it shall be created.

3.6.4.5.1.5 Transmitting Route Replies

When replying to a route request with a route reply command frame, a device that has a route discovery table entry corresponding to the source address and route request identifier of the route request SHALL construct a command frame with the frame type field set to 0x01. The source address field of the NWK header SHALL be set to the 16-bit network address of the current device and the destination address field SHALL be set to the value of the sender address field from the corresponding route discovery table entry. The device constructing the route reply SHALL populate the payload fields in the following manner.

The NWK command identifier SHALL be set to route reply.

The route request identifier field SHALL be set to the same value found in the route request identifier field of the route request command frame.

The originator address field SHALL be set to the source address in the NWK header of the route request command frame.

Using the sender address field from the route discovery table entry corresponding to the source address in the NWK header of the route request command frame, the device SHALL compute the link cost as described in section 3.6.4.1. This link cost SHALL be entered in the path cost field.

The route reply command frame is then unicast to the destination by using the MCPS-DATA.request primitive and the sender address obtained from the route discovery table as the next hop.

3.6.4.5.1.6 Transmitting Reactive Many-To-One Route Requests

See Annex K.5.3.

3.6.4.5.1.7 Route Discovery Expiration

When the route request timer expires, the device deletes the route request entry from the route discovery table. When this happens, the routing table entry corresponding to the routing address of the destination SHALL also be deleted, if its status field has a value of DISCOVERY_UNDERWAY and there are no other entries in the route discovery table created as a result of a route discovery for that destination address.

3.6.4.5.1.8 Preventing many-to-one to ad hoc route change-over

When a device receives a route request command, which is not a many-to-one route request, where a routing table entry for the destination exists, which has the many-to-one flag set to TRUE, it will not reset this flag to FALSE.

3.6.4.5.2 Upon Receipt of a Route Reply Command Frame

On receipt of a route reply command frame, a device SHALL perform the following procedure.

If the receiving device has no routing capacity it SHALL discard the command frame. Before forwarding the route reply command frame the device SHALL update the path cost field in the payload by computing the link cost from the next hop device to itself as described in section 3.6.4.1 and adding this to the value in the route reply path cost field.

To support legacy devices, a route reply received with a radius of 1 SHALL NOT be dropped. It SHALL continue to be processed as follows.

If the receiving device has routing capacity, it SHALL check whether it is the destination of the route reply command frame by comparing the contents of the originator address field of the command frame payload with its own address. If it is, it SHALL search its route discovery table for an entry corresponding to the route request identifier in the route reply command frame payload. If there is no such entry, the route reply command frame SHALL be discarded and route reply processing SHALL be terminated. If a route discovery table entry exists, the device SHALL search its routing table for an entry with a destination address field equal to the routing address corresponding to the responder address in the route reply command frame. If there is no such routing table entry, the route reply command frame SHALL be discarded and, if a route discovery table entry corresponding to the route request identifier in the route reply command frame exists, it SHALL also be removed and route reply processing SHALL be terminated. If a routing table entry and a route discovery table entry exist and if the status field of the routing table entry is set to DISCOVERY_UNDERWAY, it SHALL be changed to ACTIVE; the next hop field in the routing table SHALL be set to the previous device that forwarded the route reply command frame. The residual cost field in the route discovery table entry SHALL be set to the path cost field in the route reply payload.

If the status field was already set to ACTIVE, the device SHALL compare the path cost in the route reply command frame to the residual cost recorded in the route discovery table entry, and update the residual cost field and next hop field in the routing table entry if the cost in the route reply command frame is smaller. If the path cost in the route reply is not smaller, the route reply shall be discarded and no further processing SHALL be done. Similarly, if incoming route information is considered unsuitable as defined in section 3.6.4.5.3, the route reply shall be discarded and no further processing SHALL be done.

Note that NLDE data requests MAY be processed as soon as the first valid route is determined.

If the device receiving the route reply is not the destination, the device SHALL find the route discovery table entry corresponding to the originator address and route request identifier in the route reply command frame payload. If no such route discovery table entry exists, the route reply command frame shall be discarded. If a route discovery table entry exists, the path cost value in the route reply command frame and the residual cost field in the route discovery table entry shall be compared. If the route discovery table entry value is less than the route reply value, the route reply command frame shall be discarded. Similarly, if incoming route information is considered unsuitable as defined in section 3.6.4.5.3, the frame shall be discarded.

Otherwise, the device SHALL find the routing table entry with a destination address field equal to the routing address corresponding to the responder address in the route reply command frame. In this case, it is an error if the route discovery table entry exists and there is no corresponding routing table entry, and the route reply command frame SHOULD be discarded. The routing table entry SHALL be updated by replacing the next hop field with the address of the previous device that forwarded the route reply command frame. The route discovery table entry SHALL also be updated by replacing the residual cost field with the value in the route reply command frame.

3.6.4.5.2.1 Transmitting Relayed Route Replies

After updating its own route entry, the device SHALL forward the route reply to the destination. Before forwarding the route reply, the path cost value shall be updated. The sender SHALL find the next hop to the route reply's destination by searching its route discovery table for the entry matching the route request identifier and the source address and extracting the sender address. It SHALL use this next hop address to compute the link cost as described in section 3.6.4.1. This cost SHALL be added to the path cost field in the route reply. The destination address in the command frame NWK header SHALL be set to the next hop address and the frame SHALL be unicast to the next hop device using the MCPS-DATA.request primitive. The DstAddr parameter of the MCPS-DATA.request primitive SHALL be set to the next-hop address from the route discovery table.

If relaying the route reply failed even after exhausting all network-level retries and the reason for such failure is originating in a MAC layer status of NO_ACK, the device SHALL issue a network status command towards the responder, i.e. the originator of the route reply message, which is also the destination of the related route discovery with a status of Link Failure (0x02). It SHALL further invalidate all routing table entries that used the broken link as their next hop. The device SHALL not create a reverse routing table as described below; it MAY update such an entry if it already exists.

The NWK layer SHALL, upon relaying the route reply command frame, also create a reverse routing table entry if such an entry does not yet exist. The value of the destination address field of the routing table entry SHALL correspond to the value of the originator address field of the route reply command frame. The status field SHALL have a value of ACTIVE. The next-hop address field SHALL have a value corresponding to the next hop address in the route reply command being relayed, as determined in the previous paragraph. If the reverse routing table entry already exists the next-hop address field shall be updated, if necessary subject to suitability of incoming route information as defined in section 3.6.4.5.3.

3.6.4.5.2.2 Preventing many-to-one to ad hoc route change-over

When a device receives a route reply command, where a routing table entry for the destination exists, which has the many-to-one flag set to TRUE, it will keep the flag set to TRUE for the forward and the reverse path.

3.6.4.5.3 Assessing Suitability of Incoming Route Information

An incoming advertised route is compared to existing local routes to determine whether the advertised route is to be used to update the routing table. The incoming route information SHALL be processed as follows:

1. Search for an entry in the routing table where Destination address matches the address in the incoming route information. If no matching entry exists, one SHALL be created. Otherwise continue to step 2.
2. Check that the incoming advertised route is safe against routing loops by executing the LoopFree(R1, R2) function:

   LoopFree(R1, R2) := path-cost(R1) <= path-cost(R2)

3. LoopFree(R1, R2) verifies that a route R2 is not a sub-section of another route R1. It returns FALSE to indicate that an advertised route R1 is not to be used to update an existing route R2, as this MAY potentially cause a routing loop.

   (i) If LoopFree(advertised route, local route) returns FALSE, the advertised incoming route SHALL be ignored and SHALL NOT be used to update the routing table.

   (ii) otherwise continue to step 4.

3. Compare route costs:
   (i) If the advertised incoming route is better than the existing one, it SHALL be used to update the routing table.
   (ii) If the advertised incoming route is as good as the existing one and the existing route is ACTIVE, the incoming route SHOULD NOT be used to update the routing table, because it will offer no improvement.
   (iii) If the advertised incoming route is worse than the existing one and the existing route is ACTIVE, the incoming route SHALL NOT be used to update the routing table.

(iv) If the advertised incoming route is as good as, or worse than the existing route, and the existing route is not ACTIVE, the incoming route SHOULD be used to update the routing table, as it can safely be used to repair the existing invalid entry.

3.6.4.5.4 Calculating the Distance Between Routers and Concentrators

See Annex K.5.4.

3.6.4.5.5 Initiation and Processing of a Route Record Command Frame

If the NWK layer of a Zigbee router or Zigbee coordinator is initiating a unicast data frame as a result of an NLDE-DATA.request from the next higher layer and the many-to-one field of the routing table entry corresponding to the destination address of the frame has a value of TRUE, then the NWK layer SHALL examine the route record required field of that same routing table entry. If the route record required field also has a value of TRUE, the NWK SHALL unicast a route record command to the destination before transmitting the data frame.

If the NWK layer of a Zigbee router or Zigbee coordinator is forwarding a unicast data frame on behalf of one of its end device children and the many-to-one field of the destination's routing table entry has a value of TRUE, then the device SHALL unicast a route record command to the destination before relaying the data frame, which already contains the network short address of the Zigbee router or Zigbee coordinator in the relay list.An optional optimization is possible in which the router or coordinator MAY keep track of which of its end device children have received source routed data frames from a particular concentrator device and can thereby reduce the number of route record commands it transmits to that concentrator on behalf of its end device children.

Each relay node that receives the route record command SHALL append its network address to the command payload, increment the relay count, and forward the message. If no next hop is available, or if delivery to the next hop fails, or if there is insufficient space in the payload for the network address, the command frame shall be discarded and no error command shall be generated.

Upon receipt of the route record command by the destination, the route SHALL be stored in the source route table. Any existing source routes to the message source or intermediary nodes SHALL be replaced by the new route information.

3.6.4.6 Upon Expiration of a Route Discovery Table Entry

When a route discovery table entry is created, the expiration timer SHALL be set to expire in *nwkcRouteDiscoveryTime* OctetDurations. If the routing table entry corresponding to the source address of the route discovery table entry has any Status field value other than ACTIVE and there are no other entries in the route discovery table corresponding to that routing table entry, the routing table entry SHALL also be deleted.

3.6.4.7 Upon Expiration of a Many-To-One Route

When a routing table entry is created or updated and its many-to-one field is set to TRUE , the behavior of aging the route will depend on the presence of the Concentrator Information TLV in the Route Request. If TLV is present then the device SHALL set a timer equal to the Concentrator Discovery Time value from the TLV + nwkRouteDiscoveryTime. If the TLV is not present, or the Concentrator Discovery Time value inside the TLV is set to 0, then the timer is not set. When a many-to-one route request from the same device is received, the Expired flag is set to FALSE and the timer is reset

3.6.4.8 Route Maintenance

A device NWK layer SHALL maintain a failure counter for each neighbor to which it has an outgoing link, *i.e.*, to which it has been required to send data frames. If the outgoing link is classified as a failed link, then the device SHALL respond as described in the following paragraphs. Implementers MAY choose a simple failure-counting scheme to generate this failure counter value or they MAY use a more accurate time-windowed scheme. Note that it is important not to initiate repair too frequently since repair operations MAY flood the network and cause other traffic disruptions. Routing table entries MAY be overwritten in order to make room for new routes. Entries SHOULD be ranked by RecentActivity and TotalUsageCount fields, dropping routes that have not been used recently before dropping routes in active use; in case two routes have been used equally often recently, TotalUsageCount SHALL be considered, keeping routes that have been used more often than others, overall.

3.6.4.8.1 In Case of Link Failure

If a failed link is encountered while the device is forwarding a unicast frame using normal unicast routing, the device SHALL issue a network status command frame back to the source device of the frame with a status code of 0x02 (Link Failure) (see Table 3-52), and issue an NLME-NWK-STATUS.indication to the next higher layer with a status code indicating the reason for the failure.

Router parents will monitor route errors sent to their children and take action on behalf of them. When relaying a network status command frame by a router to its end device child that is the intended destination of the route error, where the status code field of the command frame payload has a value of 0x00, 0x01 or 0x02 indicating a link failure, the NWK layer will remove the routing table entry corresponding to the value of the destination address field of the command frame payload, if one exists. It will then relay the frame as usual to the end device.

If all attempts on all active interfaces fail while a device is forwarding a unicast data frame using a routing table entry with the many-to-one field set to TRUE and the Expired field set to FALSE, a network status command frame with status code of 0x0c indicating many-to-one route failure SHALL be generated. The destination address field in the NWK header of the network status command frame SHALL be equal to the destination address field in the NWK header of the frame causing the error. The destination address field of the network status command payload SHALL be equal to the source address field in the NWK header of the frame causing the error. The network status command frame SHALL be unicast to a random router neighbor using the MCPS-DATA.request primitive. Because it is a many-to-one route, all neighbors within concentrator radius are EXPECTED to have a routing table entry to the destination. Upon receipt of the network status command frame, if no routing table entry for the destination is present, or if delivery of the network status command frame to the next hop in the routing table entry fails, the network status command frame SHALL again be unicast to a random router neighbor using the MCPS-DATA.request primitive. The radius counter in the NWK header will limit the maximum number of times the network status command frame is relayed. Upon receipt of the network status command frame by its destination it SHALL be passed up to the next higher layer using the NLME-NWK-STATUS.indication primitive. Many-to-one routes, which have not expired, are not automatically rediscovered by the NWK layer due to route errors.

If all attempts on all active interfaces fail while a device is forwarding a unicast data frame using a routing table entry with the many-to-one field set to TRUE and the Expired field set to TRUE, the device SHALL delete the many-to-one routing table entry and MAY automatically attempt to discover an ad hoc route.

If all attempts on all active interfaces fail while the device is forwarding a unicast frame using normal unicast routing, the device SHALL issue a network status command frame back to the source device of the frame with a status code indicating the reason for the failure (see Table 3-52), and issue an NLME-NWK-STATUS.indication to the next higher layer with a status code indicating the reason for the failure.

If all attempts on all active interfaces fail while the device is forwarding a unicast frame using source routing, the device SHALL issue a network status command frame back to the source device of the frame with status code 0x0b – ‘Source route failure’, and issue an NLME-NWK-STATUS.indication to the next higher layer with the same status code of 0x0b.

On receipt of a network status command frame by a router that is the intended destination of the command where the status code field of the command frame payload has a value of 0x01 or 0x02 indicating a link failure, the NWK layer will remove the routing table entry corresponding to the value of the destination address field of the command frame payload, if one exists, and inform the next higher layer of the failure using the NLME-NWK-STATUS.indication using the same status code.

On receipt of a network status command frame by a router that is the parent of an end device that is the intended destination, where the status code field of the command frame payload has a value of 0x01 or 0x02 indicating a link failure, the NWK layer will remove the routing table entry corresponding to the value of the destination address field of the command frame payload, if one exists. It will then relay the frame as usual to the end device.

On receipt of a network status command frame by an end device, the NWK layer SHALL inform the next higher layer of the failure using the NLME-NWK-STATUS.indication.

On receipt of a network status command frame by a router that is the intended destination of the command where the status code field of the command frame payload has a value of 0x0b indicating a source route failure, the NWK layer will remove the source route corresponding to the value of the destination address field of the command frame payload, if one exists, from *nwkRouteRecordTable* and inform the next higher layer of the failure using the NLME-NWK-STATUS.indication using the same status code.

On receipt of a network status command frame by a router that is the parent of an end device that is the intended destination, where the status code field of the command frame payload has a value of 0x0b indicating a source route failure, the NWK layer will remove the source route corresponding to the value of the destination address field of the command frame payload, if one exists, from *nwkRouteRecordTable*. It will then relay the frame as usual to the end device.

If an end device encounters a failed link to its parent, the end device SHALL inform the next higher layer using the NLME-NWK-STATUS.indication primitive with a Status parameter value of 0x09 indicating parent link failure (see Table 3-52).

The APSDE of an end device MAY optionally take action when it notices an APS ACK has not been received. This option is provided for robustness in the case of intermediary devices that do not generate route errors back to the source correctly. This SHALL only be done once per APS Transaction sequence number. This SHALL NOT be done if the network destination of the APSDE transaction is the router parent. If the APSDE of an end device chooses to do this, then it SHALL do the following:

1. Initiate a Network Status Command Frame.
2. The Network destination of the message SHALL be set to the address of its router parent.
3. The Target Address in the network payload SHALL be set to the destination of the effected APSDE transaction.
4. The Status Code SHALL be set to 0x02, Link Failure.

3.6.4.8.2 Route Repair Functionality

A potentially transient routing problem is indicated via an NLME-NWK-STATUS.indication with the status set to No Route Available, Non-tree Link Failure, Source Route Failure, or Many-to-one route failure. Those error codes indicate that the problem may be overcome by repairing the route. The method for repairing the route depends upon the many-to-one field of the corresponding failed route entry in the routing table (nwkRouteTable).

1. If the route table entry indicates the many-to-one field is set to FALSE, then the following MAY be done.
    a. The stack initiates an NLME-ROUTE-DISCOVERY.req with DstAddrMode set to 0x02 (16-bit address of individual device), DstAddr set to the NetworkAddr returned by the NLME-STATUS.indication, and NoRouteCache set to TRUE.
2. If the route table entry indicates the many-to-one field is set to TRUE and the local device is a concentrator, the following MAY be done.
    a. Repair the incoming route to the concentrator. The concentrator issues NLME-NWK-ROUTE-DISCOVERY.req with the DstAddrMode set to 0x00 (No destination) and the DstAddr set to the all routers broadcast address (0xFFFC).
    b. Update the concentrator's local routing table so it may send a new message to the target destination by soliciting a message from the destination with an updated route. This can be done by the concentrator issuing a ZDO IEEE_Addr_req to the all routers broadcast address with the NWKAddrOfInterest set to the NetworkAddr returned by the NLME-STATUS.indication, and the RequestType set to 0x00 (single device response).[7]

[7] CCB 2007

3.6.5 Scheduling Beacon Transmissions

Scheduled beaconing SHALL NOT be performed in Zigbee mesh networks. IEEE Std 802.15.4 Beacon requests SHALL be answered with IEEE Std 802.15.4 beacons on demand when they are received.

3.6.6 Broadcast Communication

This section specifies how a broadcast transmission is accomplished within a Zigbee network. Any device within a network MAY initiate a broadcast transmission intended for a number of other devices that are part of the same network. A broadcast transmission is initiated by the local APS sub-layer entity through the use of the NLDE-DATA.request primitive by setting the DstAddr parameter to a broadcast address as shown in Table 3-76, or by the NWK layer through the use of these same broadcast addresses in the construction of an outgoing NWK header. (Note that broadcast transmission for link status and route request command frames is handled differently as described in section 3.6.4.4 and section 3.6.4.5.1 respectively.)

Table 3-76. Broadcast Addresses

Broadcast Address	Destination Group
0xffff	All devices in PAN
0xfffe	Reserved
0xfffd	*macRxOnWhenIdle* = TRUE
0xfffc	All routers and coordinator
0xfffb	Low power routers only
0xfff8 - 0xfffa	Reserved

To transmit a broadcast MSDU, the NWK layer of a Zigbee router or Zigbee coordinator issues an MCPS-DATA.request primitive to the MAC sub-layer(s) with the DstAddrMode parameter set to 0x02 (16-bit network address) and the DstAddr parameter set to 0xffff. For a Zigbee end device, the MAC destination address of the broadcast frame SHALL be set equal to the 16-bit network address of the parent of the end device. The PANId parameter SHALL be set to the PANId of the Zigbee network. This specification does not support broadcasting across multiple networks. Broadcast transmissions SHALL NOT use the MAC sub-layer acknowledgement; instead, a passive acknowledgement mechanism SHALL be used. Passive acknowledgement means that every Zigbee router and Zigbee coordinator keeps track of which of its neighboring devices have successfully relayed the broadcast transmission. The MAC sub-layer acknowledgement is disabled by setting the acknowledged transmission flag of the TxOptions parameter to FALSE. All other flags of the TxOptions parameter SHALL be set based on the network configuration.

The Zigbee coordinator, each Zigbee router and those Zigbee end devices with *macRxOnWhenIdle* equal to TRUE, SHALL keep a record of any new broadcast transaction that is either initiated locally or received from a neighboring device. This record is called the broadcast transaction record (BTR) and SHALL contain at least the sequence number and the source address of the broadcast frame. The broadcast transaction records are stored in the *nwkBroadcastTransactionTable* (BTT) as shown in Table 3-77.

Table 3-77. Broadcast Transaction Record

Field Name	**Size**	**Description**
Source Address	2 bytes	The 16-bit network address of the broadcast initiator.
Sequence Number	1 byte	The NWK layer sequence number of the initiator's broadcast.
Expiration Time	1 byte	A countdown timer indicating the number of seconds until this entry expires; the initial value is *nwkNetworkBroadcastDeliveryTime.*

Processing of a broadcast with a NWK source of the local device SHALL only be done when the device has been powered up and operating on the network for nwkNetworkBroadcastDeliveryTime. This prevents broadcasts from being processed that might have recently originated from the device after a reset.

When a device receives a broadcast frame from a neighboring device, it SHALL compare the destination address of the frame with its device type. If the destination address does not correspond to the device type of the receiver as outlined in Table 3-76, the frame shall be discarded. If the destination address corresponds to the device type of the receiver, the device SHALL compare the sequence number and the source address of the broadcast frame with the records in its BTT.

If the device has a BTR of this particular broadcast frame in its BTT, it MAY update the BTR to mark the neighboring device as having relayed the broadcast frame. It SHALL then drop the frame. If no record is found, it SHALL create a new BTR in its BTT and MAY mark the neighboring device as having relayed the broadcast. The NWK layer SHALL then indicate to the higher layer that a new broadcast frame has been received using the NLDE-DATA.indication. If the device is a Zigbee router (ZR) or a Zigbee Coordinator (ZC) and the radius field is greater than zero; then the frame shall be retransmitted. Otherwise it shall be dropped. Before the retransmission, it SHALL wait for a random time period called broadcast jitter. This time period shall be bounded by the value of the *nwkcMaxBroadcastJitter* attribute. Zigbee end devices with *macRxOnWhenIdle* equal to FALSE SHALL NOT participate in the relaying of broadcast frames and need not maintain a BTT for broadcast frames that they originate.

If, on receipt of a broadcast frame, the NWK layer finds that the BTT is full and contains no expired entries, then the frame SHOULD be dropped. In this situation the frame SHOULD NOT be retransmitted, nor SHOULD it be passed up to the next higher layer.

A Zigbee coordinator or Zigbee router operating in a non-beacon-enabled Zigbee network SHALL retransmit a previously broadcast frame at most *nwkMaxBroadcastRetries* times. If the device does not support passive acknowledgement, then it SHALL retransmit the frame exactly *nwkMaxBroadcastRetries* times. If the device supports passive acknowledgement and any of its neighboring devices have not relayed the broadcast frame within *nwkPassiveAckTimeout* OctetDurations then it SHALL continue to retransmit the frame on the MAC interfaces which are in communication with such neighbors up to a maximum of *nwkMaxBroadcastRetries* times.

A device SHOULD change the status of a BTT entry after *nwkNetworkBroadcastDeliveryTime* OctetDurations have elapsed since its creation. The entry status SHOULD change to expired and thus the entry can be overwritten if required when a new broadcast is received.

A router or coordinator with the *macRxOnWhenIdle* MAC PIB attribute set to TRUE, which has one or more neighbors with the *macRxOnWhenIdle* MAC PIB attribute set to FALSE, SHALL do the following:

1. If the NWK destination address of the broadcast is 0xFFFF, for each nwkNeighborTable entry where the device type is 0x02 (Zigbee End Device) and the entry has RxOnWhenIdle =FALSE, the broadcast SHALL be re-transmitted as follows:
    a. If the end device target is NOT the NWK source of the broadcast, the message SHALL be relayed as a unicast at the MAC layer via an MCPS-DATA.request.
    b. If the end device target is the NWK source of the broadcast, no MCPS-DATA.request is generated.

   c. Go to the next applicable entry.

2. For end device neighbors where RxOnWhenIdle = TRUE, and for all router and coordinator neighbors, they will receive the message when it is re-broadcast as described in the section above.[8]

Every Zigbee router SHALL have the ability to buffer at least 1 frame at the NWK layer in order to facilitate retransmission of broadcasts.

Figure 3-52 shows a broadcast transaction between a device and two neighboring devices**.**

[8] CCB 2231

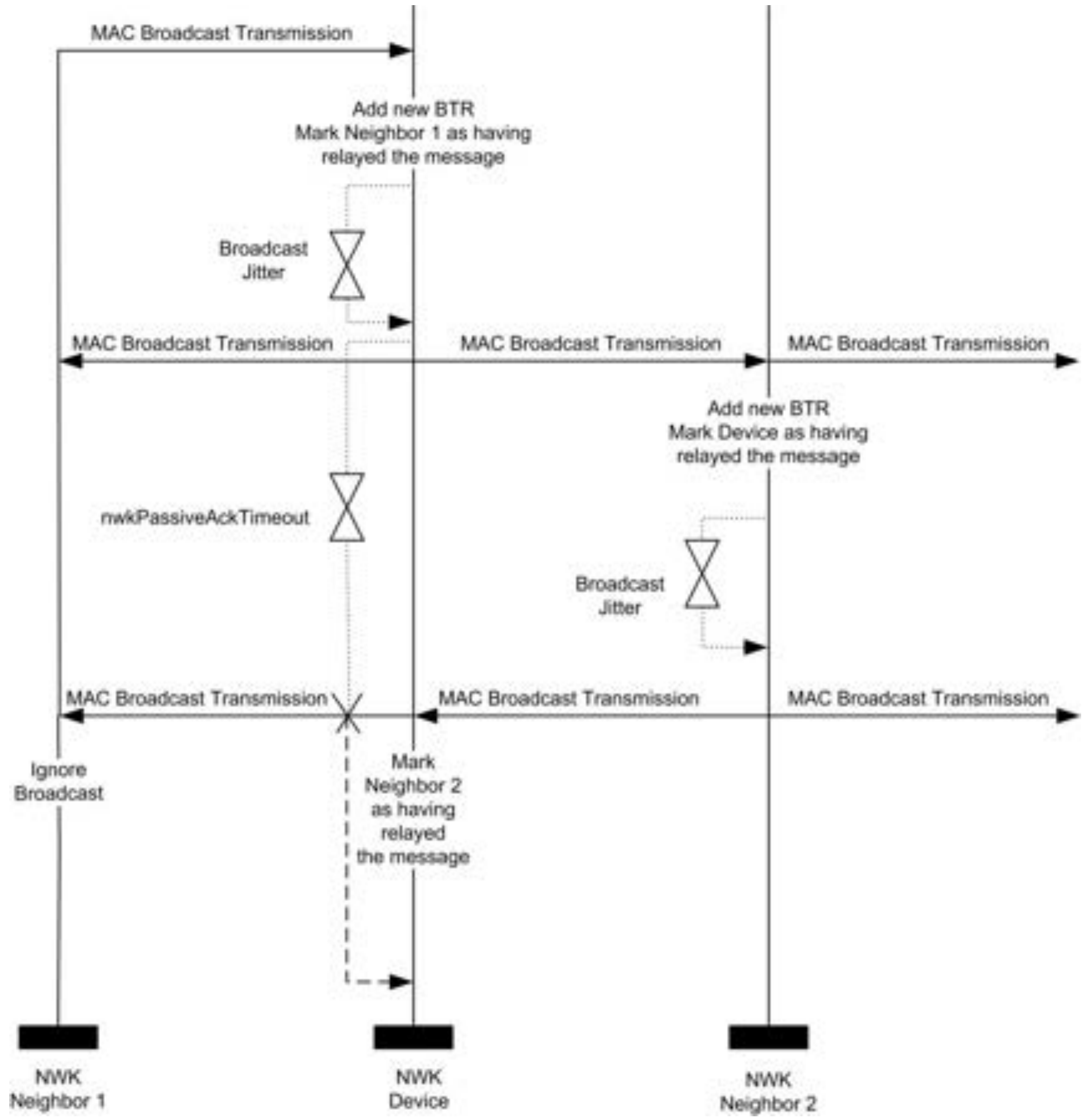


Figure 3-52. Broadcast Transaction Message Sequence Chart

3.6.7 Multicast Communication

Multicast communication is carried out using APS layer group addressing and a broadcast addressing mode at the network layer. Network-level multicast, available as an optional feature in prior revisions of this specification, is deprecated.

3.6.8 NWK Information in the MAC Beacons

This section specifies how the NWK layer uses the beacon payload of a MAC sub-layer beacon frame to convey NWK layer-specific information to neighboring devices.

Starting with this version of the specification, Zigbee defines a mechanism to append to the beacon in a future compatible way. This Beacon Appendix is included in addition to the Beacon Payload previously defined in the Zigbee specification. Together the Zigbee Beacon Info Field and the Beacon Appendix constitute the IEEE Std 802.15.4 Beacon Payload.

The Beacon Appendix SHALL constitute the entire Beacon Payload field for classic beacons. For Enhanced beacons, the the Beacon Info Field is omitted from the Beacon Payload Field and instead, it is conveyed inside a Payload Information Element. The Payload Information Element is detailed in Annex D, section Zigbee Payload IE. Table 3-78 summarizes the Beacon Payload fields.

Table 3-78. Beacon Payload Fields

	Name	Size	Description
IEEE Beacon Payload	Zigbee Beacon Info Field	15 bytes	The standard Zigbee beacon payload that has been used in R22 and all prior versions of the specification. This is defined in section 3.6.7.
	Zigbee Beacon Appendix	Variable	A set of TLVs indicating various information about the network and the local device sending the beacon. This field is only present in Revision 23 devices and later.

The Zigbee beacon Info Field SHALL contain the information shown in Table 3-79. This enables the NWK layer to provide additional information to new devices that are performing network discovery and allows these new devices to more efficiently select a network and a particular neighbor to join. Refer to section 3.6.1.6.1 for a detailed description of the network discovery procedure.

Table 3-79. Zigbee Beacon Info Fields

Name	Type	Valid Range	Description
Protocol ID	Integer	0x00 – 0xff	This field identifies the network layer protocols in use and, for purposes of this specification, SHALL always be set to 0, indicating the Zigbee protocols. The value 0xff SHALL also be reserved for future use by the Connectivity Standards Alliance.
Stack profile	Integer	0x00 – 0x0f	A Zigbee stack profile identifier.
nwkcProtocolVersion	Integer	0x00 – 0x0f	The version of the Zigbee protocol.
Router capacity	Boolean	TRUE or FALSE	This value is set to TRUE if this device is capable of accepting join requests from router-capable devices and is set to FALSE otherwise. This value SHALL match the value of RoutersAllowed for the MAC interface that this beacon is being sent from.
Device depth (DEPRECATED)	Integer	0x00	This value has been DEPRECATED.
End device capacity	Boolean	TRUE or FALSE	This value is set to TRUE if the device is capable of accepting join requests from end devices seeking to join the network and is set to FALSE otherwise.

Name	Type	Valid Range	Description
nwkExtendedPANId	64-bit extended address	0x0000000000000001 – 0xfffffffffffffffe	The globally unique ID for the PAN of which the beaconing device is a member. By default, this is the 64-bit IEEE address of the Zigbee coordinator that formed the network, but other values are possible and there is no required structure to the address.
TxOffset	Integer	0x000000 – 0xffffff	This value indicates the difference in time, measured in symbols, between the beacon transmission time of the device and the beacon transmission time of its parent; This offset MAY be subtracted from the beacon transmission time of the device to calculate the beacon transmission time of the parent. This parameter is set to the default value of 0xFFFFFF in beaconless networks.
nwkUpdateId	Integer	0x00 – 0xFF	This field reflects the value of *nwkUpdateId* from the NIB.

The NWK layer of the Zigbee coordinator SHALL update the beacon info fields immediately following network formation. All other Zigbee devices SHALL update it immediately after the join is completed and any time the network configuration changes. The beacon payload is written into the MAC sub-layer PIB using the MLME-SET.request primitive. The *macBeaconPayloadLength* attribute is set to the length of the beacon payload, and the octet sequence representing the beacon payload is written into the *macBeaconPayload* attribute. The formatting of the bit sequence representing the beacon payload is shown in Figure 3-53.

Bits: 0–7	**8–11**	**12–15**	**16–17**	**18**	**19–22**	**23**	**24–87**	**88–111**	**112–119**
Protocol ID	Stack profile	*nwk cProtocol Version*	Reserved	Router capacity	Device depth	End device capacity	*nwk Extended PANId*	Tx Offset	*Nwk UpdateId*

Figure 3-53. Format of the Zigbee Beacon Info Fields

3.6.8.1 Zigbee Beacon Appendix

This field is only present in IEEE Beacons transmitted by Revision 23 and later devices. Devices from Revision 22 and earlier are expected to accept beacons with this field and ignore the data. Revision 23 devices SHALL accept beacons without this field.

The Zigbee Beacon Appendix MAY contain one or more TLVs as defined by this specification. Unknown TLVs MAY also be included in the Beacon Appendix Payload and SHALL not generate an error on reception; they SHALL be silently ignored.

The set of TLVs included in the Beacon Appendix SHALL be set according to the values of the *nwkNetworkWideBeaconPayloadTLVs* and *nwkDeviceBeaconPayloadTLVs* of the NIB. The nwkDeviceBeaconPayloadTLVs

SHALL contain at a minimum the Router Information Global TLV. See Steps for Constructing the IEEE Std 802.15.4 Beacon for how these are utilized.

3.6.8.2 Steps for Constructing the IEEE Std 802.15.4 Beacons

These steps apply both to standard IEEE Std 802.15.4 Beacons and Enhanced Beacons.

1. Append the IEEE Std 802.15.4 MAC headers for the beacon.
2. Append the Zigbee Beacon Info field.
3. Examine each complete TLV of the *nwkNetworkWideBeaconAppendixTLVs* NIB value.
    a. If the full contents of the current TLV do **not** fit within the IEEE Std 802.15.4 frame then beacon construction is complete. No more steps are executed.
    b. If the full contents of the TLV fit within the IEEE Std 802.15.4 frame, append the complete TLV.
    c. Continue processing TLVs within the *nwkNetworkWideBeaconAppendixTLVs* NIB value.
4. Examine each complete TLV of the *nwkDeviceBeaconPayloadTLVs* NIB value.
    a. If the full contents of the current TLV do **not** fit within the IEEE Std 802.15.4 frame then beacon construction is complete. No more steps are executed.
    b. If the TLV Tag ID has been previously encountered in step 4, skip this TLV.
        i. Globally set TLVs override locally set ones.
    c. If the full contents of the TLV fit within the IEEE Std 802.15.4 frame AND the TLV ID was NOT previously appended in step 4, append the complete TLV.
    d. Continue processing TLVs within the *nwkDeviceBeaconPayloadTLVs* NIB value.

3.6.9 Persistent Data

Devices operating in the field MAY be restarted either manually or programmatically by maintenance personnel, or MAY be restarted accidentally for any number of reasons, including localized or network-wide power failures, battery replacement during the course of normal maintenance, impact, and so on.

The following information SHOULD be preserved across resets in order to maintain an operating network:

- The device's PAN Id and Extended PAN Id.
- The device's 16-bit network address.
- nwkUpdateId - The value identifying a snapshot of the network settings with which this node is operating with.

For each device in the nwkNeighborTable of the NIB with a device type set to 0x02 (Zigbee End Device), the following SHALL be saved:

- The 64-bit IEEE address
- 16-bit network address
- The End Device Configuration value
- Device Timeout value
- MAC Interface Index
- If the device is an end device, the *nwkParentInformation* value in the NIB.
- For end devices, the 16-bit network address of the parent device.
- The stack profile in use.
- The MAC Interface Table
- If the device is the Zigbee coordinator or a Zigbee router, its routing sequence number

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

3.6.10 End Device Aging and Management

The end device and router relationship is established via MAC association or NWK rejoin, and can be dissolved via a leave command. However there are a number of ways in which the relationship can get broken, where router parent and end device do not agree. For example the router parent MAY think it is still the router parent for an end device when in fact the end device has switched to a new parent, or the router parent MAY age out the child since it has had no communication with it for an extended period of time.

Router parents have a finite amount of local resources to store end device information. As such it is desirable to clean out old entries to allow for new end devices to join. End devices SHALL be aged out by the router according to the rules defined below.

Note: For *nwkParentInformation*, see Table 3-62.

3.6.10.1 End Device Aging Mechanism

A router parent SHALL age neighbor table entries for end devices. It is important to note that prior versions of this specification did not have this requirement and thus legacy devices exist that do not have this child aging mechanism.

A router parent SHALL keep track of the amount of real time that has passed and decrement the Timeout counter value for each end device entry in its neighbor table until the value reaches 0. When a neighbor table entry's Timeout counter value reaches 0, the router parent SHALL delete the entry from the neighbor table.

End Devices MAY periodically send a keepalive message to reset the Timeout counter value. See section 3.6.10.3 for details.

3.6.10.2 Establishing the Timeout

A router SHALL initially set the timeout for all end devices according to the default value of *nwkEndDeviceTimeoutDefault* in Table 3-62.

The following describes how an end device MAY update this value from the default.

After joining or rejoining the network the end device SHALL send an End Device Timeout Request command to its parent. This SHALL be done even if the end device is joining or rejoining to the same parent. The message SHALL include their timeout period and configuration.

Routers SHALL process the End Device Timeout Request command as follows:

1. If the Requested Timeout Enumeration value in the frame is not within the valid range, it SHALL generate an End Device Timeout Response command with a status of INCORRECT_VALUE and no further processing of the message SHALL be done.
2. The parent SHALL find the neighbor table entry for the sending device and verify that the entry corresponds to an end device. If no entry is found or the entry is not an end device, then the message SHALL be dropped and no further processing SHALL be done.
3. The parent SHALL validate that each bit set to 1 in the End Device Configuration Field is a known feature and supported by the parent. If any feature is not supported or not known to the parent, it SHALL send an End Device Timeout Response with a status of UNSUPPORTED_FEATURE and no further processing SHALL be done. At the time of this specification Revision, there are no defined bits for the End Device Configuration Field, and thus no supported features
4. The received value SHALL be converted into an actual timeout amount. This SHALL be done by obtaining the actual timeout value for the corresponding Requested Timeout Enumeration in Table 3-57. The value SHALL be converted from minutes into seconds if it is not already a value in seconds. The parent SHALL set the Timeout Counter and Device Timeout values of the neighbor table entry to the converted value.
5. The parent SHALL set the End Device Configuration information in the neighbor table for the corresponding end device's entry to the value of the End Device Configuration field in the received message.

6. The parent SHALL generate an End Device Timeout Response command with a status of SUCCESS. It SHALL fill in the value of the *Parent Information Bitmask* field according to the keepalive methods it supports. The parent SHALL set either Mac Data Poll Keepalive Support or End Device Timeout Request Support. The parent is indicating the method that the End Device SHALL use.

An End Device that receives an End Device Timeout Response Command SHALL process it as follows.

1. If the status is SUCCESS it SHALL set the *nwkParentInformation* value in the NIB to value of the Parent Information field of the received command. No further processing SHALL be done.
2. If the End Device receives the command with a status value other than SUCCESS, it SHALL assume its timeout value has not been configured on the parent.

End Devices MAY receive no End Device Timeout Response command at all if they are communicating with a legacy device that does not have support for this command. They SHALL treat this the same as receiving an End Device Timeout Response with a non-SUCCESS status code.

3.6.10.3 End Device Keepalive

All end devices (including RxOnWhenIdle=TRUE) that have received an End Device Timeout Response Command with a status of SUCCESS MAY periodically send a keepalive to their router parent to insure they remain in the router's neighbor table.

The keepalive message will refresh the timeout on the parent device so that the parent does not delete the child from its neighbor table. The period for sending the keepalive to the router parent SHALL be determined by the manufacturer of the device and is not specified by this standard. It is recommended that the period allows the end device to send 3 keepalive messages during the Device Timeout period. This will help insure that a single missed keepalive message will not age out the end device on the router parent.

There are two keepalive mechanisms described below. The method the end device uses depends on the support of the router parent. The router parent will indicate its support in the End Device Timeout Response command frame and this information will be stored in the NIB.

When an End Device needs to send a keepalive message, it SHALL examine the *nwkParentInformation* value in the NIB. If bit 0 has a value of 1 (indicating support of the MAC data poll keepalive) then the device SHALL send a MAC data poll command unicast to its parent.

Otherwise if the value of bit 1 has a value of 1, then the device SHALL send an End Device Timeout Request command as a unicast to refresh the keepalive timer. If the transmission is successful, the device SHALL wait for *macResponseWaitTime* for an End Device Timeout Response from its parent. If the transmission was unsuccessful, or if no End Device Timeout Response command is received, or if the status field indicates a value other than SUCCESS, the end device SHALL generate a NLME-NWK-STATUS.indication with a code of 0x09 (Parent Link Failure).

3.6.10.4 MAC Data Poll Processing

A router whose *nwkParentInformation* in the NIB has bit 1 set to 0, SHALL support the MAC Data poll as an End Device keepalive. A router is not required to support this method. If it does not it SHALL support the End Device Timeout Request method.

Upon receipt of an MLME-POLL.Indication the router parent SHALL examine its neighbor table and do **one** of the following:

1. If there is no entry in the neighbor table corresponding to the DeviceAddress of the MLME-Poll.Indication primitive, then the device SHALL construct a leave message. The destination NWK address SHALL be set to the value of the MAC source of the MAC data poll. See section 3.6.10.4.1 for more information on the leave message. The message SHALL be added to the indirect transaction queue of the MAC layer. No further processing shall be done.
2. If there is an entry in the neighbor table for the sending device's MAC source, then the local device shall set the Timeout counter value to the value of the *End Device Keepalive Timeout* value, and it SHALL set the Keepalive Received value to TRUE.

When an End Device sends a MAC Data poll command it SHALL assume that the parent has knowledge of the end device and the Timeout Counter associated with the end device has been reset in the parent's neighbor table. The End Device will behave per reference [B1] with regard to the data pending bit in the MAC ACK, and will follow standard processing of any leave message that MAY be received after sending a data poll.

A router SHALL only update the Keepalive Received value on receipt of an MLME-POLL.indication when the *nwkParentInformation* has bit 0 set to 1.

3.6.10.4.1 Sending a Leave Message

A router SHALL send a leave message when it wants to inform an end device it is no longer a parent to the end device. The leave message SHALL be one of the following messages:

1. NWK Leave Request
    a. A device that chooses to send a NWK leave request SHALL set fields of the NWK Command as follows.
        i. The destination IEEE address sub-field of the frame control SHALL be set to 0, indicating that no destination IEEE address is present.
        ii. The destination IEEE address field SHALL NOT be present in the message.
        iii. The request sub-field of the command options field SHALL be set to 1.
        iv. The rejoin request sub-field of the command SHALL be set to 1.
2. ZDO Mgmt_Leave_req
    a. A device that chooses to send a ZDO Mgmt_Leave_req SHALL set the fields of the of the ZDO Mgmt_leave_req command as follows:
        i. The Device Address field SHALL be set to NULL (0x0000000000000000)
        ii. The Remove Children Bit SHALL be set to 0.
        iii. The Rejoin bit SHALL be set to 1.
    b. The Acknowledgement request sub-field of the APS Frame control field SHALL be set to 0 (no acknowledgement requested).

3.6.10.5 Setting the End Device Timeout on the Router Parent

A router SHALL set the default values for Timeout Counter and End Device Keepalive Timeout to the time-span indicated by *nwkEndDeviceTimeoutDefault* as converted to seconds.

After successfully joining or rejoining the network and receiving the network key, an End Device SHALL send an End Device Timeout Request command to its router parent indicating its desired timeout. Upon receipt and successful processing of the End Device Timeout Request router parents SHALL update the timeout values accordingly. See section 3.6.10.2 for details.

Legacy devices will not send an End Device Timeout Request and thus will receive the default timeout.

3.6.10.6 Local End Device Timeout

An end device MAY keep track of its timeout using the following mechanism:

1. The end device SHALL find the corresponding neighbor table entry for its router parent.
2. It SHALL decrement the Timeout Counter value in the Neighbor Table entry based on the amount of real time that has passed, until that value reaches 0.
3. If the Timeout Counter reaches a value of 0, it SHALL assume that its parent has timed out the device.

If the end device has determined that it has been timed out, it can choose to perform a rejoin to get back on the network as described in section 3.6.1.6.1. Alternatively it is permissive for an end device to always perform a rejoin without keep tracking of its local end device timeout.

There is no requirement that the end device re-establish connectivity with the network if it has determined that it has reached the timeout value established with its router parent. An end device MAY choose to delay rejoining the network until it is appropriate, for example when the end device has data it needs to send.

3.6.10.7 Persistent Values on the Parent Router

The router parent is EXPECTED to persistently store the end device information in the neighbor table (see section).

3.6.10.8 Reboot and Child Aging

On reboot routers SHALL set the Timeout Counter value for each end device in its neighbor table to the entry's value of Device Timeout. In other words, end devices SHALL be given a full time period for aging out.

On reboot it is recommended end devices immediately initiate a keep-alive message to verify connectivity to their parent.

3.6.10.9 Diagrams Illustrating End Device Management

Figure 3-54 shows an end device joining into a network and the series of message exchanges. After the end device has joined and has a copy of the NWK key, it will send a NWK command of End Device Request to the parent and check for a response.

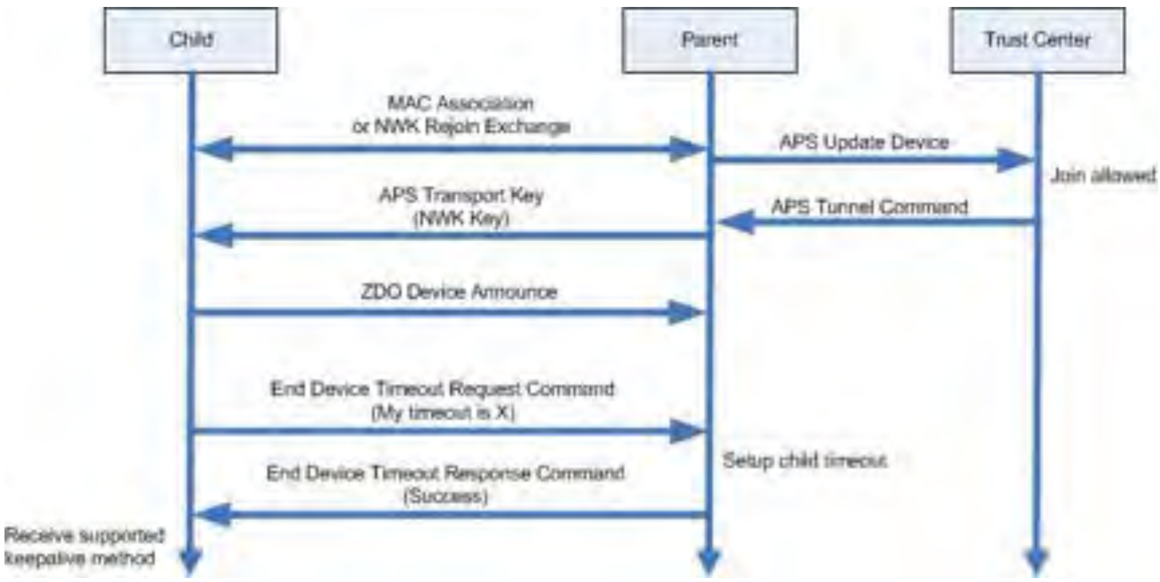


Figure 3-54. Initial Setup of the End Device Timeout

Figure 3-55 shows normal operation of a child talking to a parent that supports the MAC Data Poll Keepalive Method. When the data pending bit is unset in the MAC acknowledgement, the end device can assume that the parent still remembers the device.

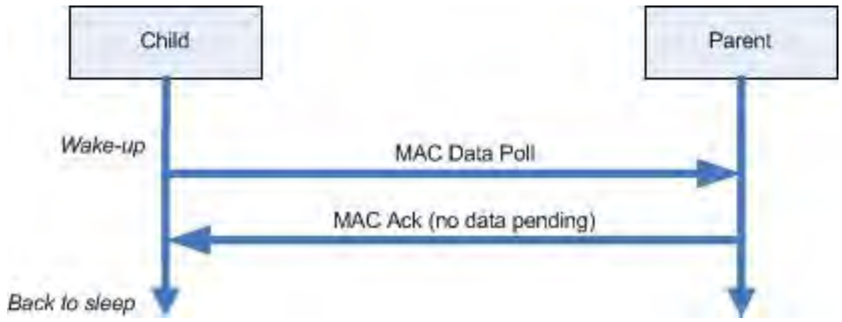


Figure 3-55. Child Keepalive: MAC Data Poll Method

Figure 3-56 shows normal operation of a child talking to a parent that supports the End Device Timeout Request keepalive method.

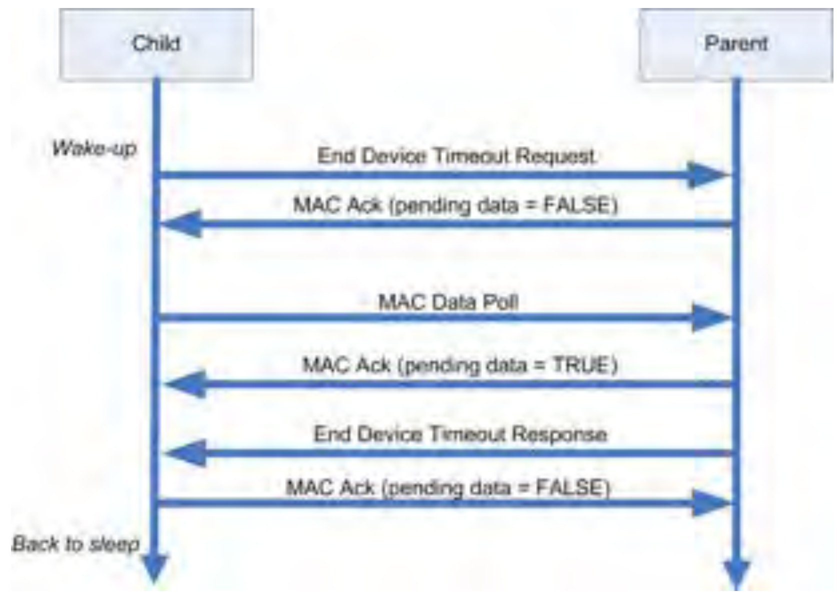
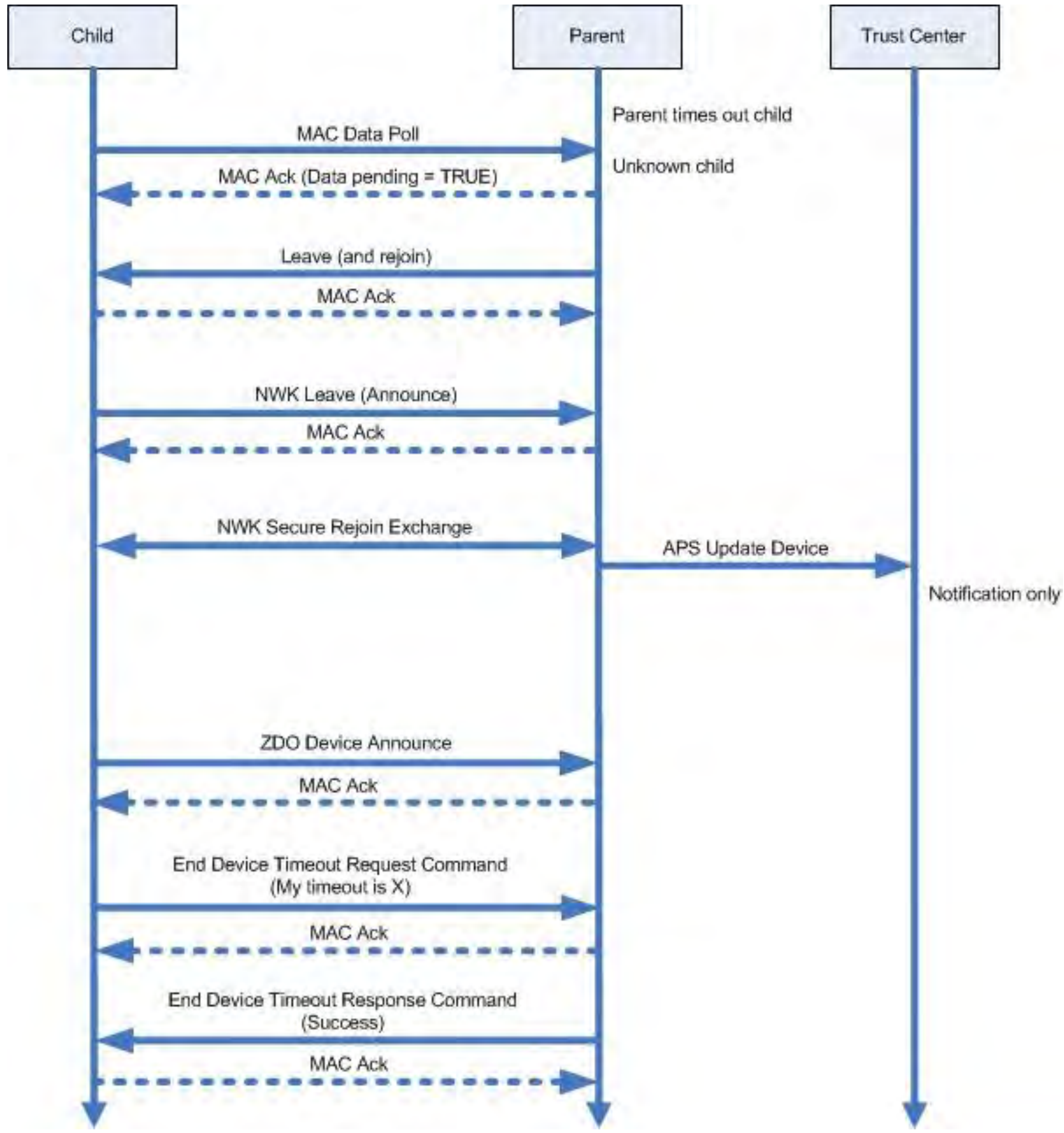


Figure 3-56. Child Keepalive: End Device Timeout Request Method

Figure 3-57 and Figure 3-58 show what happens when a parent that supports the MAC data poll keepalive method, ages out the child. The parent will indicate to the child that it has a pending message for the child by setting the data pending bit to TRUE in the MAC acknowledgement. The parent will then transmit a leave message to the device with the rejoin bit set to TRUE. The device will announce leaving the network and perform a rejoin. shows a secure rejoin while shows a Trust Center Rejoin. After the rejoin is successful the device will send the NWK Command End Device Timeout Request and receive a response.

9617 

9618 **Figure 3-57. Aging out Children: MAC Data Poll Method - Secure Rejoin**

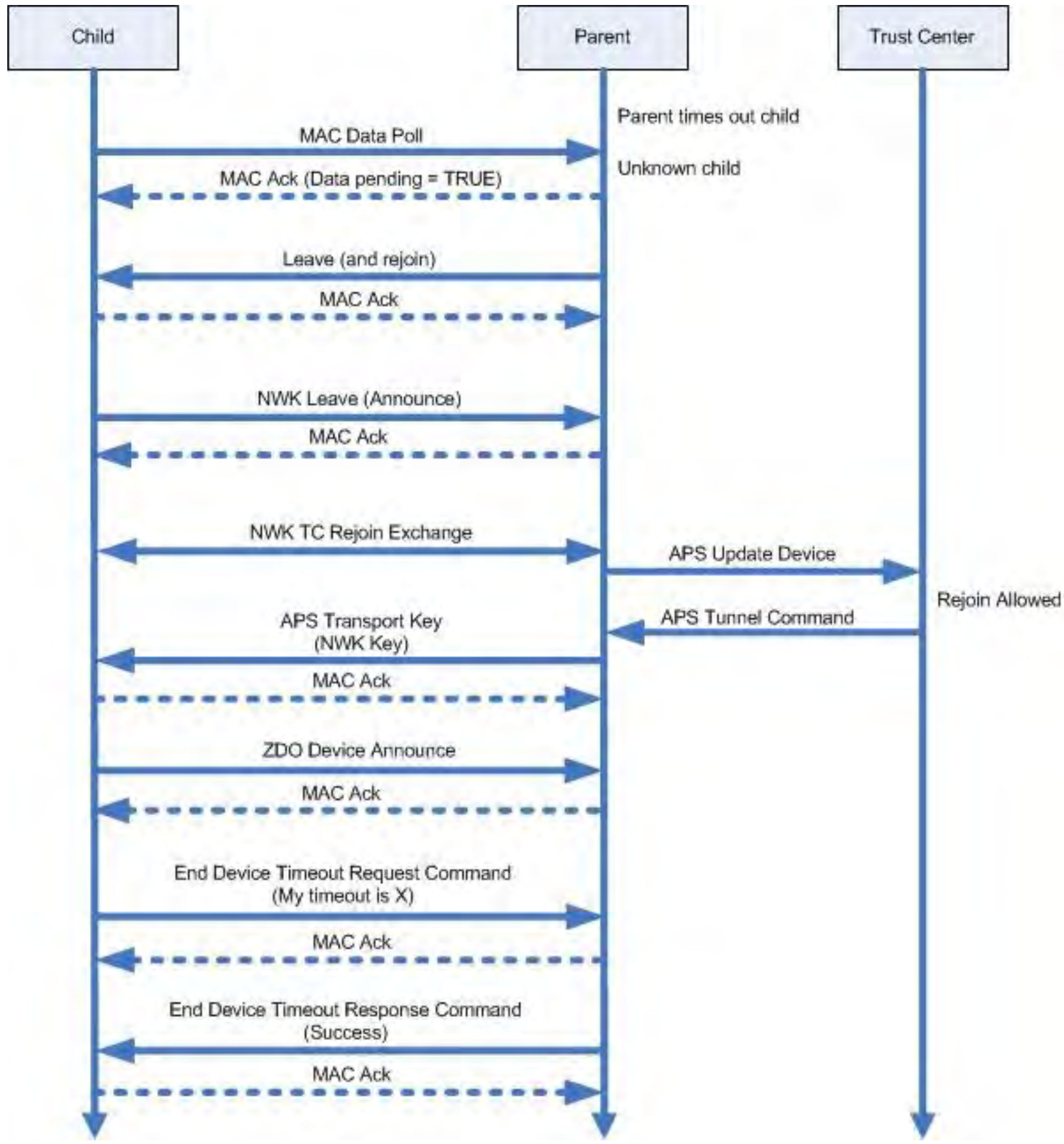
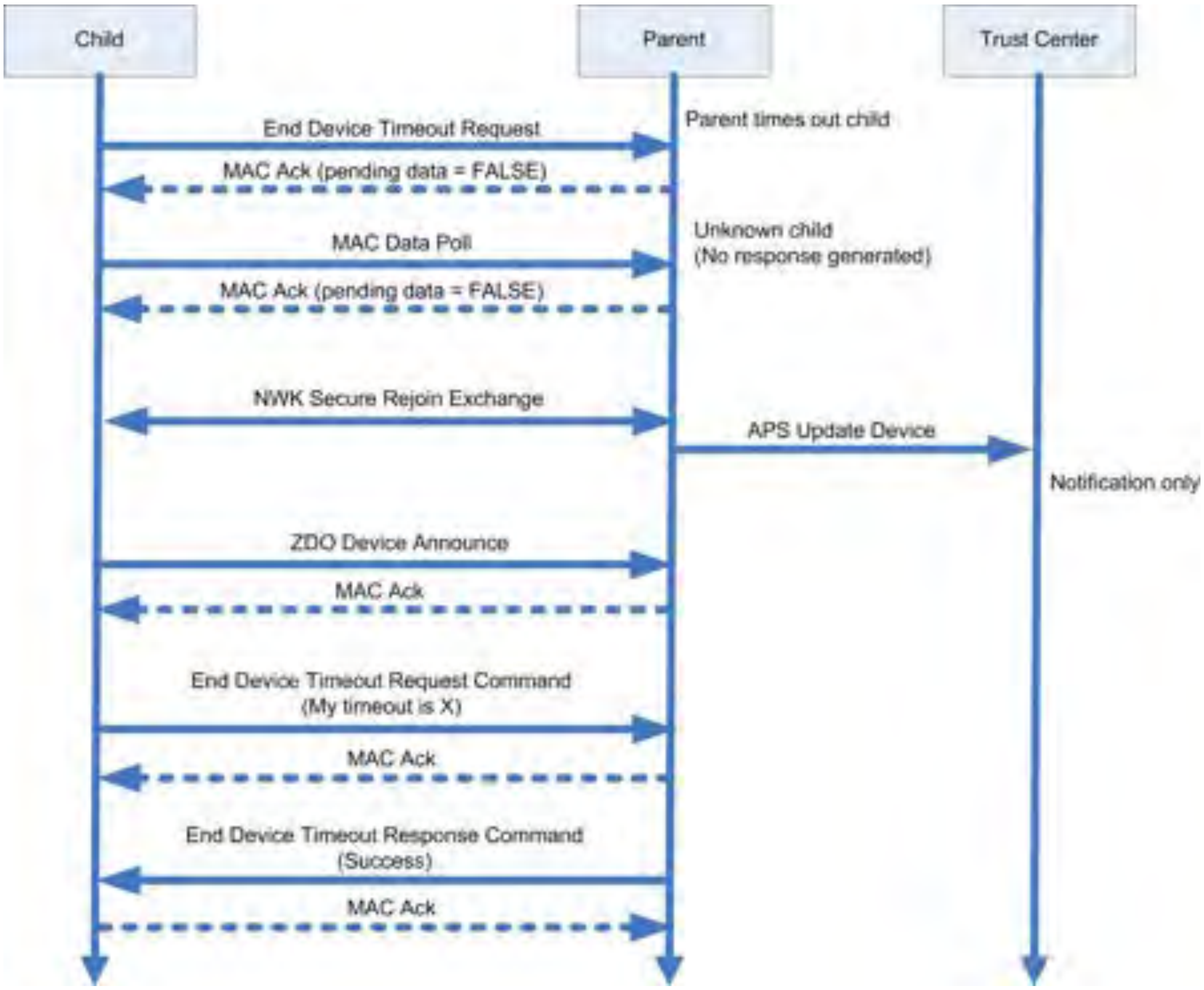


Figure 3-58. Aging out Children: MAC Data Poll - Trust Center Rejoin

Figure 3-59 and Figure 3-60 show what happens when an end device is aged out of the parent's table with a parent that supports the End Device Timeout Request method. An end device sends an End Device Timeout Request and receives no response. Afterwards it will perform a rejoin. shows a secure rejoin while shows a Trust Center rejoin. Once the device has completed the rejoin it will send a NWK command End Device timeout request and receive the response.

9626 

9627 **Figure 3-59. Aging out Children: End Device Timeout Request Method - Secure Rejoin**

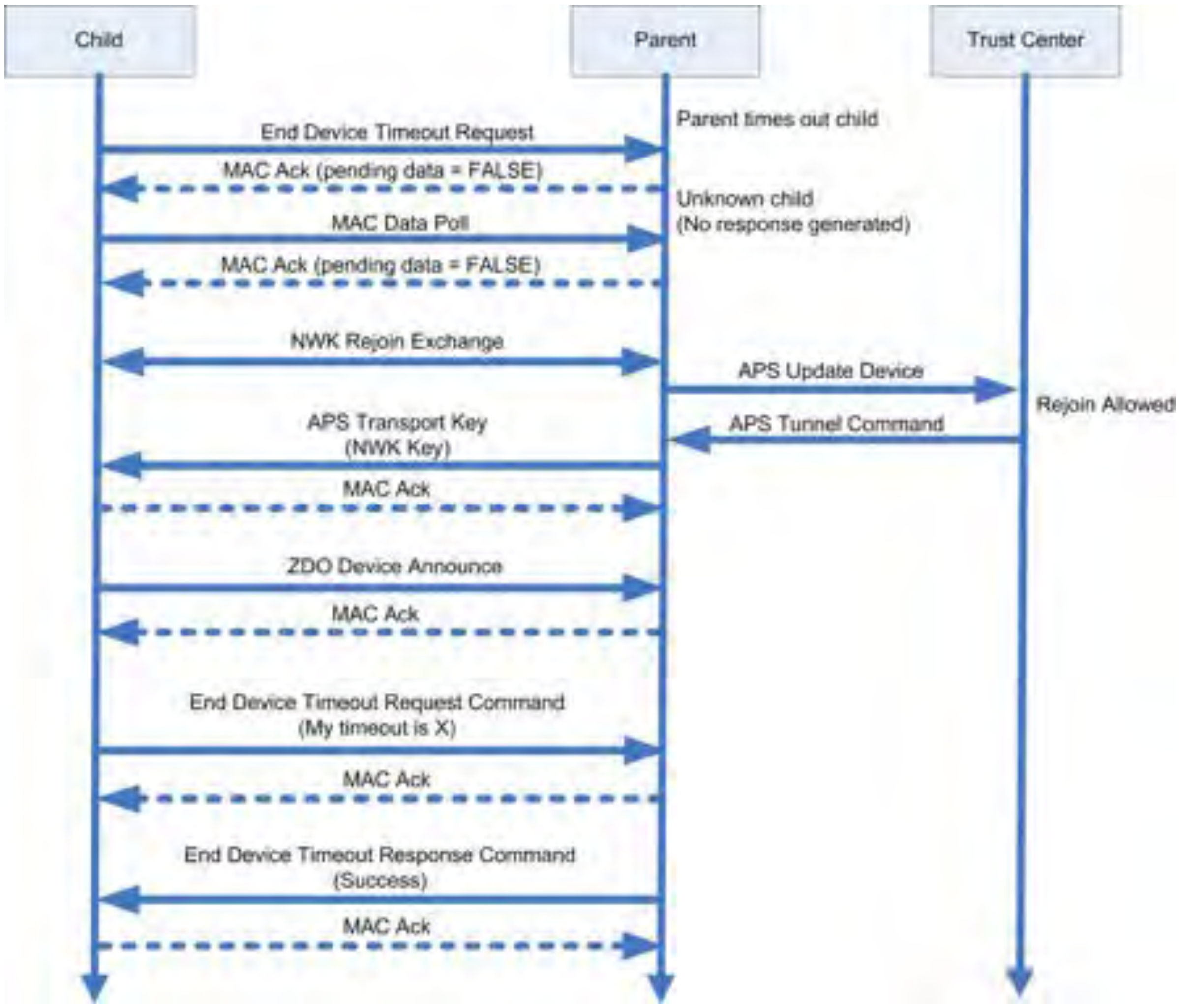


Figure 3-60. Aging out Children: End Device Timeout Request Method - Trust Center Rejoin

3.6.10.10 Trust Center Rejoin or Secure Rejoin

An end device that has detected it has been aged out of its parent's child table MAY choose to use either a Secure Rejoin or a Trust Center rejoin. The choice to use one or the other is up to the implementation but can be based on whether it MAY have missed a network key update. A device that has missed a network key update will have to use a Trust Center Rejoin. However in a case where that situation has not occurred, a Secure Rejoin will complete more quickly and can be used instead. It is possible that an end device MAY try both methods to insure it can get back on the network.

3.6.11 Power Negotiation

Two devices can negotiate a lower power level than the device's normal operating power. This is considered a "good neighbor" policy of reducing transmission noise beyond what is necessary for the two devices to communicate. Power negotiation is an optional feature that a device MAY choose to implement. This feature is described in detail in section 3.4.13.

3.6.11.1 Behavior

Each device SHALL maintain a table of devices and the required power level for communicating to those devices. This table is contained within the MLME. See section D.11.2. The network layer SHALL use that table for negotiating power levels with neighboring devices.

If there is a change in network channel, or a device performs a rejoin, the maximum allowed power SHALL be used initially.

3.6.11.2 Determining support for Power Negotiation

It is preferable that an End Device does not have to generate periodic Link Power Delta commands unless its parent supports the Power Negotiation feature. Therefore it can utilize the Parent Information field of the End Device Timeout response to discern whether the parent device supports this. If the parent indicates bit 2 (Power Negotiation Support) is set to 0, the End Device SHALL set the *nwkLinkPowerDeltaTransmitRate* to 0. If the parent indicates bit 2 (Power Negotiation Support) is set to 1, the End Device SHALL set the *nwkLinkPowerDeltaTransmitRate* to an appropriate rate based its desired balance of battery life versus latency of renegotiating its power level.

3.6.11.2.1 Generating Link Power Delta messages

A router SHALL generate a Link Power Delta message as follows:

1. The message SHALL be broadcast to all non-sleeping end devices, router and coordinator devices (0xFFFD).
2. The radius of the broadcast SHALL be 1.

After generating a Link Power Delta command an end device SHALL poll up to 3 times in rapid succession to receive a corresponding Link Power Delta command from the parent.

3.6.12 Multiple MAC Interfaces

It is permissible that the NLME supports multiple MAC interfaces. This MAY be done in order to support multiple potential PHY where only a single PHY is enabled at one time, or it MAY be to support simultaneous operation on multiple PHYs.

The following describes the procedure for devices supporting multiple MAC.

3.6.12.1 Multi-MAC Selection Device

A Multi-MAC selection device is one that supports multiple possible MACs, but only operates on one while it is joined to a network. The process for selecting among the multiple MACs to use for joining is the following:

1. For each possible interface, the following is executed.
    a. Application enables one of the MAC interfaces by issuing the NLME-SET-INTERFACE.request primitive with State = TRUE. For all other interfaces, it issues the NLME-SET-INTERFACE.request primitive with State = FALSE.
    b. Application issues the NLME-GET-INTERFACE.request and waits for the NLME-GET-INTERFACE.confirm with a Status of SUCCESS.
    c. Application issues the NLME-JOIN.request primitive with the ScanChannelsList set to the value returned in the SupportedChannels parameter of the NLME-GET-INTERFACE.confirm.
    d. If NLME-JOIN.confirm returns a result of SUCCESS, then the Multi-PHY Selection is complete.
    e. If NLME-JOIN.confirm returns a result other than SUCCESS, than proceed to the next interface.

3.6.12.2 Multi-MAC Switch Device

A Multi-MAC Switch Device is a device that supports simultaneous operation on multiple MACs and will bridge packets from one interface to the other. The following is the procedure for a device to enable multiple MACs.

1. The application can tailor the list of interfaces that are enabled by selecting a subset of channels that only a single interface supports.
2. The Application issues an NLME-NETWORK-FORMATION.request with the ScanChannelsList and MacInterfaceIndex set to the SupportedChannels and the InterfaceIndex indicated by the parameters of the NLME-GET-INTERFACE.confirm primitive.

   a. For each interface, the NLME-NETWORK-FORMATION.request calls the NLME-SET-INTERFACE.request with a status of Enabled

3. When the device chooses to enable additional MAC interfaces it SHALL do the following for each interface.
   a. Enable the interface entry by issuing an NLME-SET-INTERFACE.request primitive and wait for the NLME-SET-INTERFACE.confirm primitive.
   b. Retrieve the list of supported channels by issuing NLME-GET-INTERFACE.request primitive and waiting for the NLME-GET-INTERFACE.confirm primitive.
   c. Issue the NLME-NETWORK-AND-PARENT-DISCOVERY.request with the ScanChannelList set to the SupportedChannelList indicated by the NLME-GET-INTERFACE.confirm primitive.
4. If any of the interfaces returned the NLME-NETWORK-AND-PARENT-DISCOVERY.confirm primitive with a NetworkDescriptor containing a PANId that is the same as the nwkPANId of the NIB, the following SHALL be done to change the PAN ID.
   a. Randomly select a new PAN ID and set the nwkPanId value of the NIB to this new value.
   b. Issue a Network Update Command with the Update Command ID set to 0x00, PAN ID Update, and the Update Count set to 1. The New PAN ID field SHALL be set to the nwkPanId value of the NIB.

3.6.13 Unknown Commands

Note that devices conforming to R21 or earlier of this specification will not return this response to an unknown command.

Whenever an unknown or unsupported unicast NWK command is received and the NWK Destination of the frame is the address of the local device, it SHALL do the following:

1. Construct a Network Status Command Frame
2. Set the Status field of the command to 0x13, Unknown Command.
3. Set the Destination Address field of the command to the NWK Source of the frame.
4. Set the Payload of the command as shown in Table 3-80, with the Command ID set to the ID value of the received command.

Octets: 1
Command ID

Figure 3-61. Unknown Command Payload

5. Issue a NLDE-DATA.request containing the Network Status Command frame. The NWK Destination SHALL be equal to NWK Source of the frame that triggered this behavior.

3.7 NWK Layer Status Values

Network (NWK) layer confirmation primitives often include a parameter that reports on the status of the request to which the confirmation applies. Values for NWK layer Status parameters appear in Table 3-80.

Table 3-80. NWK Layer Status Values

Name	Value	Description
SUCCESS	0x00	A request has been executed successfully.
Reserved	0x01 – 0xc0	Reserved for future use.
INVALID_PARAMETER	0xc1	An invalid or out-of-range parameter has been passed to a primitive from the next higher layer.
INV_REQUESTTYPE	0xc2	The next higher layer has issued a request that is invalid or cannot be executed given the current state of the NWK layer.
NOT_PERMITTED	0xc3	An NLME-JOIN.request has been disallowed.
STARTUP_FAILURE	0xc4	An NLME-NETWORK-FORMATION.request has failed to start a network.
ALREADY_PRESENT	0xc5	A device with the address supplied to the NLME-ADD-NEIGHBOR.request is already present in the neighbor table of the device on which the NLME-ADD-NEIGHBOR.request was issued.
SYNC_FAILURE	0xc6	Used to indicate that an NLME-SYNC.request has failed at the MAC layer.
NEIGHBOR_TABLE_FULL	0xc7	An NLME-JOIN-DIRECTLY.request has failed because there is no more room in the neighbor table.
UNKNOWN_DEVICE	0xc8	An NLME-LEAVE.request has failed because the device addressed in the parameter list is not in the neighbor table of the issuing device.
UNSUPPORTED_ATTRIBUTE	0xc9	An NLME-GET.request or NLME-SET.request has been issued with an unknown attribute identifier.
NO_NETWORKS	0xca	An NLME-JOIN.request has been issued in an environment where no networks are detectable.
Reserved	0xcb	Reserved for future use.
MAX_FRM_COUNTER	0xcc	Security processing has been attempted on an outgoing frame, and has failed because the frame counter has reached its maximum value.
NO_KEY	0xcd	Security processing has been attempted on an outgoing frame, and has failed because no key was available with which to process it.

Name	**Value**	**Description**
BAD_CCM_OUTPUT	0xce	Security processing has been attempted on an outgoing frame, and has failed because the security engine produced erroneous output.
Reserved	0xcf	Reserved for future use.
ROUTE_DISCOV-ERY_FAILED	0xd0	An attempt to discover a route has failed due to a reason other than a lack of routing capacity.
ROUTE_ERROR	0xd1	An NLDE-DATA.request has failed due to a routing failure on the sending device or an NLME-ROUTE-DISCOV-ERY.request has failed due to the cause cited in the accompa-nying NetworkStatusCode.
BT_TABLE_FULL	0xd2	An attempt to send a broadcast frame has failed because there is no room in the BTT.
FRAME_NOT_BUFFERED	0xd3	An NLDE-DATA.request has failed due to insufficient buff-ering available.
INVALID_INTERFACE	0xd5	An attempt was made to use a MAC Interface with a state that is currently set to FALSE (disabled) or that is unknown to the stack..
MISSING_TLV	0xD6	A required TLV for processing the request was not present.
INVALID_TLV	0xD7	A TLV was malformed or missing relevant information.

CHAPTER 4. SECURITY SERVICES SPECIFICATION

4.1 Document Organization

The remaining portions of this document specify in greater detail the various security services available within the Zigbee stack. Basic definitions and references are given in section 4.2. A general description of the security services is given in section 4.2.1. In this section, the overall security architecture is discussed; basic security services provided by each layer of this architecture are introduced. Sections 4.2.2 and 4.2.3 give the Connectivity Standards Alliance's security specifications for the Network (NWK) layer and the Application Support Sublayer (APS) layer, respectively. These sections introduce the security mechanisms, give the primitives, and define any frame formats used for security purposes. Section 4.5 describes security elements common to the NWK and APS layers. Section 4.6 provides a basic functional description of the available security features. Finally, annexes provide technical details and test vectors needed to implement and test the cryptographic mechanisms and protocols used by the NWK and APS layers.

4.2 General Description

Security services provided for Zigbee include methods for key establishment, key transport, frame protection, and device management. These services form the building blocks for implementing security policies within a Zigbee device. Specifications for the security services and a functional description of how these services SHALL be used are given in this document.

4.2.1 Security Architecture and Design

In this section, the security architecture is described. Where applicable, this architecture complements the security services that are already present in the IEEE Std 802.15.4-2020 [B1] security specification.

4.2.1.1 Security Assumptions

The level of security provided by the Zigbee security architecture depends on the safekeeping of the symmetric keys, on the protection mechanisms employed, and on the proper implementation of the cryptographic mechanisms and associated security policies involved. Trust in the security architecture ultimately reduces to trust in the secure initialization and installation of keying material and to trust in the secure processing and storage of keying material.

Implementations of security protocols, such as key establishment, are assumed to properly execute the complete protocol and not to leave out any steps thereof. Random number generators are assumed to operate as EXPECTED. Furthermore, it is assumed that secret keys do not become available outside the device in an unsecured way. That is, a device will not intentionally or inadvertently transmit its keying material to other devices unless the keying material is protected, such as during key-transport. During initial key transport the keying material used for protection MAY be a well-known key, thus resulting in a brief moment of vulnerability where the key could be obtained by any device. Alternatively, the initial key transport MAY be done using a pre-shared secret key that is passed out-of-band from the Zigbee network. The following caveat in these assumptions applies: due to the low-cost nature of *ad hoc* network devices, one cannot generally assume the availability of tamper-resistant hardware. Hence, physical access to a device MAY yield access to secret keying material and other privileged information, as well as access to the security software and hardware.

Due to cost constraints, Zigbee has to assume that different applications using the same radio are not logically separated (for example, by using a firewall). In addition, from the perspective of a given device it is not even possible (barring certification) to verify whether cryptographic separation between different applications on another device — or even between different layers of the communication stack thereof — is indeed properly implemented. Hence, one SHALL assume that separate applications using the same radio trust each other; that is, there is no cryptographic task separation. Additionally, lower layers (for example, APS, NWK, or MAC) are fully accessible by any of the applications. These assumptions lead to an open trust model for a device; different layers of the communication stack and all applications running on a single device trust each other.

In summary:

- The provided security services cryptographically protect the interfaces between different devices only.
- Separation of the interfaces between different stack layers on the same device is arranged non-cryptographically, via proper design of security service access points.

4.2.1.2 Security Design Choices

The open trust model (as described in section 4.2.1.1) on a device has far-reaching consequences. It allows re-use of the same keying material among different layers on the same device and it allows end-to-end security to be realized on a device-to-device basis rather than between pairs of particular layers (or even pairs of applications) on two communicating devices.

However, one SHALL also take into consideration whether one is concerned with the ability of malevolent network devices to use the network to transport frames across the network without permission.

These observations lead to the following architectural design choices:

First, the principle that "*the layer that originates a frame is responsible for initially securing it*" is established. For example, if a NWK command frame needs protection, NWK layer security SHALL be used.

Second, if protection from theft of service is required (that is, from malevolent network devices), NWK layer security SHALL be used for all frames, except those passed between a router and a newly joined device (until the newly joined device receives the active network key). Thus, only a device that has joined the network and successfully received the active network key will be able to have its messages communicated more than one hop across the network.

Third, due to the open trust model, security can be based on the reuse of keys by each layer. For example, the active network key SHALL be used to secure APS layer broadcast frames or NWK layer frames. Reuse of keys helps reduce storage costs.

Fourth, end-to-end security is provided such that it is possible for only source and destination devices to access messages protected by a shared key. This ensures that routing of messages between the two devices with the shared key can be independent of trust considerations.

Fifth, to simplify interoperability of devices, the base security level used by all devices in a given network, and by all layers of a device, SHALL be the same. If an application needs more security for its payload than is provided by network level security, it can establish application level security with another device. There are several policy decisions which any real implementation SHALL address correctly. Application profiles SHOULD include policies to:

Handle error conditions arising from securing and unsecuring packets. Some error conditions MAY indicate loss of synchronization of security material, or MAY indicate ongoing attacks:

- Detect and handle loss of counter synchronization and counter overflow.
- Detect and handle loss of key synchronization.
- Expire and periodically update keys, if desired.

The other security design choice is done by the device that forms a network. This device sets the security policies and processes followed by the network and devices that join the network.

4.2.1.2.1 Security Keys

Security amongst a network of Zigbee devices is based on "link" keys and a "network" key. Unicast communication between APL peer entities is secured by means of a 128-bit link key shared by two devices, while broadcast communications and any network layer communications are secured by means of a 128-bit network key shared amongst all devices in the network. The intended recipient is always aware of the exact security arrangement; that is, the recipient knows whether a frame is protected with a link key or a network key.

A device SHALL acquire link keys either via key-transport, or pre-installation (for example, during factory installation). A device SHALL acquire a network key via key-transport. Some application profiles have also developed out of band mechanisms or key negotiation protocols used for generating link keys or network keys on devices. Ultimately, security between devices depends on secure initialization and installation of these keys.

There is one type of network key; however, it can be used in either distributed or centralized security models. The security model controls how a network key is distributed; and MAY control how network frame counters are initialized. The security model does not affect how messages are secured.

There are two different types of trust center link keys: global and unique. The type of trust center link key in use by the local device SHALL determine how the device handles various trust center messages (APS commands), including whether to apply APS encryption. A Trust Center link key MAY also be used to secure APS data messages between the Trust Center and the corresponding peer device. The choice of whether to use APS security on those APS data messages is up to the higher layer application.

A link key between two devices, neither of which is the trust center, is known as an application link key.

The default value for the centralized security global trust center link key SHALL have a value of 5A 69 67 42 65 65 41 6C 6C 69 61 6E 63 65 30 39 (ZigbeeAlliance09).

The different types of keys used are described in Table 4-1.

Table 4-1. Link Keys Used in Zigbee Networks

Key Name	Description
Centralized security global trust center link key	Link key used for joining centralized security networks.
Distributed security global link key	Link key used for joining distributed security networks.
Install code link key	Link key derived from install code from joining device to create unique trust center link key for joining.
Application link key	Link key used between two devices for application layer encryption.
Device Specific trust center link key	Link key used between the trust center and a device in the network. Used for trust center commands and application layer encryption.

In a secured network there are a variety of security services available. Prudence dictates that one would prefer to avoid re-use of keys across different security services, which otherwise could cause security leaks due to unwanted interactions. As such, these different services use a key derived from a one-way function using the link key (as specified in section 4.5.3). The use of uncorrelated keys ensures logical separation of the execution of different security protocols. The key-load key is used to protect transported link keys; the key-transport key is used to protect transported network keys. The active network key MAY be used by the NWK and APL layers of Zigbee. As such, the same network key and associated outgoing and incoming frame counters SHALL be available to all of these layers. The link keys MAY be used only by the APS sublayer. As such, the link key SHALL be available only to the APL layer.

An installation code is a short code that uses an algorithm to derive the 128-bit AES key. The mechanism for deriving a key from an installation code are out of scope of this specification.

4.2.1.2.2 Zigbee Security Architecture

The Zigbee security architecture includes security mechanisms at two layers of the protocol stack. The NWK and APS layers are responsible for the secure transport of their respective frames. Furthermore, the APS sublayer provides services for the establishment and maintenance of security relationships. The Zigbee Device Object (ZDO) manages the security policies and the security configuration of a device. Figure 1-1 shows a complete view of the Zigbee protocol stack. The security mechanisms provided by the APS and NWK layers are described in this version of the specification.

4.2.2 NWK Layer Security

When a frame originating at the NWK layer needs to be secured Zigbee SHALL use the frame-protection mechanism given in section 4.3.1 of this specification, unless the SecurityEnable parameter of the NLDE-DATA.request primitive is FALSE, explicitly prohibiting security. For example, no NWK layer security is used during transport of the NWK Key over the last hop to a joining device since APS security will be used to protect the frame. The NWK layer's frame-protection mechanism SHALL make use of the Advanced Encryption Standard (AES) [B8] and use CCM* as specified in Annex A. The security level applied to a NWK frame SHALL be determined by the *nwkSecurityLevel* attribute in the NIB. Upper layers manage NWK layer security by setting up active and alternate network keys and by determining which security level to use.

Figure 4-1 shows an example of the security fields that MAY be included in a NWK frame.

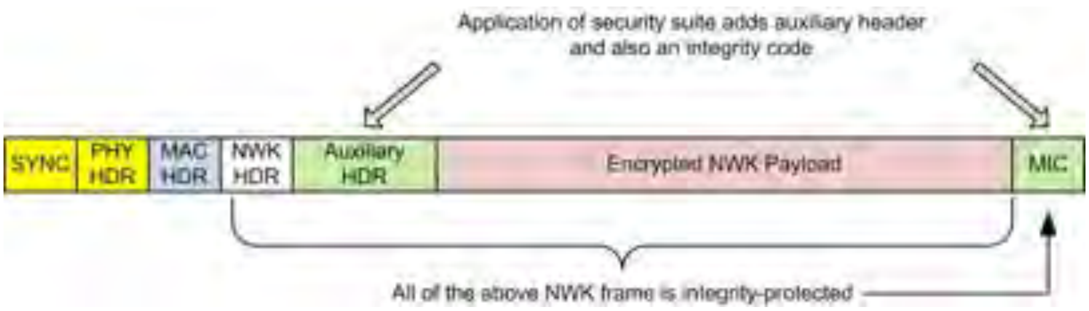


Figure 4-1. Zigbee Frame with Security on the NWK Level

4.2.3 APL Layer Security

When a frame originating at the APL layer needs to be secured, the APS sublayer SHALL handle security. The APS layer's frame-protection mechanism is given in section 4.4.1 of this specification. The APS layer allows frame security to be based on link keys or the network key. Figure 4-2 shows an example of the security fields that MAY be included in an APL frame. The APS layer is also responsible for providing applications and the ZDO with key establishment, key transport, and device management services.

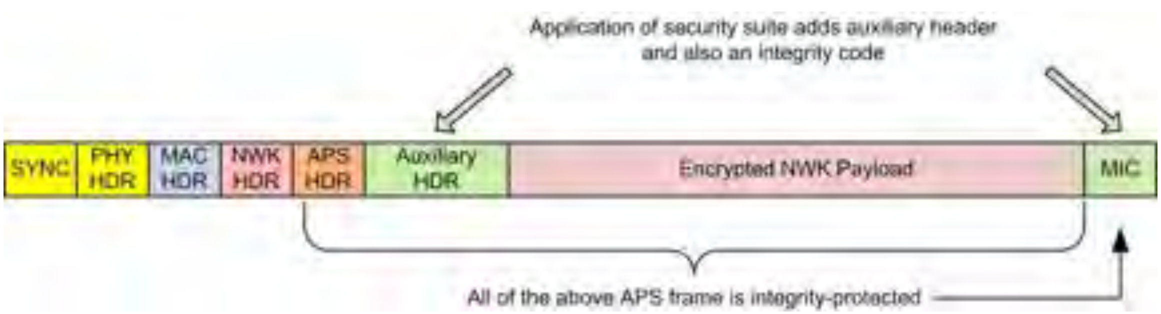


Figure 4-2. Zigbee Frame with Security on the APS Level

4.2.3.1 Transport Key

The transport-key service provides secured means to transport a key to another device or other devices. The secured transport-key command provides a means to transport link, or network key from a key source (for example, the Trust Center) to other devices.

4.2.3.2 Update Device

The update device service provides a secure means for a router device to inform the Trust Center that a third device has had a change of status that SHALL be updated (for example, the device joined or left the network). This is the mechanism by which the Trust Center maintains an accurate list of active network devices.

4.2.3.3 Remove Device

The remove device service provides a secure means by which a Trust Center informs a router device that one of the router's children or the router itself SHALL be removed from the network. For example, the remove device service MAY be employed to remove from a network a device that has not satisfied the Trust Center's security requirements for network devices.

4.2.3.4 Request Key

The request-key service provides a secure means for a device to request an end-to-end application link key or trust center link key, from the Trust Center.

4.2.3.5 Switch Key

The switch-key service provides a secure means for a Trust Center to inform another device that it SHOULD switch to a different active network key.

4.2.3.6 Verify-Key

The verify-key service provides a secure means for a device to verify that the device and the Trust Center agree on the current value of the device's link key.

4.2.3.7 Confirm Key

The confirm-key service provides a secure means for a Trust Center to confirm a previous request to verify a link key.

4.2.4 Trust Center Role

For security purposes, Zigbee defines the role of "Trust Center". The Trust Center is the device trusted by devices within a network to distribute keys for the purpose of network and potentially end-to-end application configuration management. All members of the network SHALL recognize exactly one active Trust Center, and there SHALL be exactly one Trust Center in each centralized security network. The Trust Center is responsible for establishing, maintaining and updating security policies for the network.

In a distributed security network, all routers have the capability to act as the Trust Center and distribute keys for network security. This distributed trust center role is used for network key distribution but not trust center link key distribution since there is not a singular trust center in the network.

In some applications a device can be pre-loaded with the Trust Center address and initial Trust Center link key, or the joining device's Trust Center link key can be installed out of band.

In applications that can tolerate a moment of vulnerability, the network key can be sent via APS secured key transport using a well-known link key.

In a centralized security model, the Trust Center established policies for joining devices and network security. It MAY require devices to be known before providing the network key update for joining, or MAY require a preconfigured link key be installed out of band. These Trust Center policies are described in section 4.7.1.

In a centralized security network a device securely communicates with its Trust Center using the current Trust Center link key.

For purposes of trust management, a device only accepts a Trust Center link key or active network key originating from its Trust Center via key transport. For purposes of network management in a centralized security network, a device accepts an initial active network key and updated network keys only from its Trust Center when secured with its Trust Center Link key. For purposes of configuration, a device accepts link keys intended for establishing end-to-end security between two devices only from its Trust Center or through application level negotiation using a higher level protocol between the two devices. Aside from the initial Trust Center link key or network key, additional link, and network keys are only accepted if they originate from a device's Trust Center via secured key transport or negotiated using higher level application protocols.

4.3 NWK Layer Security

The NWK layer is responsible for the processing steps needed to securely transmit outgoing frames and securely receive incoming frames. Upper layers control the security processing operations by setting up the appropriate keys and frame counters and establishing which security level to use.

4.3.1 Frame Security

The detailed steps involved in security processing of outgoing and incoming NWK frames are described in sections 4.3.1.1 and 4.3.1.2, respectively.

4.3.1.1 Security Processing of Outgoing Frames

If the NWK layer has a frame, consisting of a header *NwkHeader* and payload *Payload*, which needs security protection and *nwkSecurityLevel* > 0, and in the case of a NWK data frame, the SecurityEnabled parameter in NLD-EDATA.request had a value of TRUE, it SHALL apply security as follows:

1) Obtain the *nwkActiveKeySeqNumbe*r from the NIB and use it to retrieve the active network key *key*, outgoing frame counter *OutgoingFrameCounter*, and key sequence number *KeySeqNumber* from the *nwkSecurityMaterialSet* attribute in the NIB. Obtain the security level from the *nwkSecurityLevel* attribute from the NIB. If the outgoing frame counter is equal to 2^{32}-1, or if the key cannot be obtained, security processing SHALL fail and no further security processing SHALL be done on this frame.
2) Construct the auxiliary header *AuxiliaryHeader* (see section 4.5.1):
    a) Set the security control field as follows:
        i) The security level sub-field SHALL be the security level obtained from step 1.
        ii) The key identifier sub-field SHALL be set to ‘01’ (that is, the active network key).
        iii) The extended nonce sub-field SHALL be set to 1.
    b) Set the source address field to the 64-bit extended address of the local device.
    c) Set the frame counter field to the outgoing frame counter from step 1.
    d) Set the key sequence number field to the sequence number from step 1.
3) Execute the CCM mode encryption and authentication operation, as specified in Annex A, with the following instantiations:
    a) Obtain the parameter *M* from Table 4-38 corresponding to the security level from step 1.
    b) The bit string *Key* SHALL be the key obtained from step 1.
    c) The nonce *N* SHALL be the 13-octet string constructed using the security control field from step a, the frame counter field from step d, and the source address field from step c (see section 4.5.2.2).
    d) If the security level requires encryption, the octet string *a* SHALL be the string *NwkHeader* || *AuxiliaryHeader* and the octet string *m* SHALL be the string *Payload*. Otherwise, the octet string *a* SHALL be the string *NwkHeader* || *AuxiliaryHeader* || *Payload* and the octet string *m* SHALL be a string of length zero.
4) If the CCM mode invoked in step 3 outputs ‘invalid’, security processing SHALL fail and no further security processing shall be done on this frame.
5) Let *c* be the output from step 3. If the security level requires encryption, the secured outgoing frame SHALL be *NwkHeader* || *AuxiliaryHeader* || *c*, otherwise the secured outgoing frame SHALL be *NwkHeader* || *AuxiliaryHeader* || *Payload* || *c*.

6) If the secured outgoing frame size is greater than *aMaxMacFrameSize* security processing SHALL fail and no further security processing SHALL be done on this frame.

7) The outgoing frame counter from step 1 SHALL be incremented by one and stored in the *OutgoingFrame-Counter* element of the network security material descriptor referenced by the *nwkActiveKeySeqNum-ber* in the NIB; that is, the outgoing frame counter value associated with the key used to protect the frame is updated.

8) The security level sub-field of the security control field SHALL be overwritten by the 3-bit all-zero string '000'.

4.3.1.2 Security Processing of Incoming Frames

If the NWK layer receives a secured frame (consisting of a header *NwkHeader*, auxiliary header *AuxiliaryHeader*, and payload *SecuredPayload*) as indicated by the security sub-field of the NWK header frame control field, it SHALL perform security processing as follows:

1) Determine the security level from the *nwkSecurityLevel* attribute of the NIB. Over-write the 3-bit security level sub-field of the security control field of the *AuxiliaryHeader* with this value. Determine the sequence number *SequenceNumber*, sender address *SenderAddress*, and received frame count *ReceivedFrameCount* from the auxiliary header *AuxiliaryHeader* (see section 4.5.1). If *ReceivedFrameCounter* is equal to 2^{32}-1, security processing SHALL indicate a failure to the next higher layer and no further security processing shall be done on this frame.

2) Obtain the appropriate security material (consisting of the key and other attributes) by matching *SequenceNumber* to the sequence number of any key in the *nwkSecurityMaterialSet* attribute in the NIB. If the security material cannot be obtained, security processing SHALL indicate a failure to the next higher layer with a status of 'frame security failed' and no further security processing shall be done on this frame.

3) If there is an incoming frame count *FrameCount* corresponding to *SenderAddress* from the security material obtained in step 2 and if *ReceivedFrameCount* is less than *FrameCount*, security processing SHALL indicate a failure to the next higher layer with a status of 'bad frame counter' and no further security processing shall be done on this frame.

4) Execute the CCM mode decryption and authentication checking operation, as specified in section A.2, with the following instantiations:

   a) The parameter *M* SHALL be obtained from Table 4-38 corresponding to the security level from step 1.

   b) The bit string *Key* SHALL be the key obtained from step 2.

   c) The nonce *N* SHALL be the 13-octet string constructed using the security control, the frame counter, and the source address fields from *AuxiliaryHeader* (see section 4.5.1). Note that the security level subfield of the security control field has been overwritten in step 1 and now contains the value determined from the *nwkSecurityLevel* attribute from the NIB.

   d) The octet string *SecuredPayload* SHALL be parsed as *Payload1* || *Payload2*, where the rightmost string *Payload2* is an *M*-octet string. If this operation fails, security processing SHALL indicate a failure to the next higher layer with a status of 'frame security failed' and no further security processing shall be done on this frame.

   e) If the security level requires decryption, the octet string *a* shall be the string *NwkHeader* || *AuxiliaryHeader* and the octet string *c* SHALL be the string *SecuredPayload*. Otherwise, the octet string *a* SHALL be the string *NwkHeader* || *AuxiliaryHeader* || *Payload1* and the octet string *c* SHALL be the string *Payload2*.

5) Return the results of the CCM operation:

   a) If the CCM mode invoked in step 4 outputs 'invalid', security processing SHALL indicate a failure to the next higher layer with a status of 'frame security failed' and no further security processing shall be done on this frame.

   b) Let *m* be the output of step 4. If the security level requires encryption, set the octet string *Unsecured-NwkFrame* to the string *NwkHeader* || *m*. Otherwise, set the octet string *UnsecuredNwkFrame* to the string *NwkHeader* || *Payload1*.

6) Set *FrameCount* to (*ReceivedFrameCount* + 1) and store both *FrameCount* and *SenderAddress* in the NIB. If storing this frame count and address information will cause the memory allocation for this type of information

to be exceeded, and the *nwkAllFresh* attribute in the NIB is TRUE, then security processing SHALL fail and no further security processing shall be done on this frame. *UnsecuredNwkFrame* now represents the unsecured received network frame and security processing SHALL succeed. So as to never cause the storage of the frame count and address information to exceed the available memory, the memory allocated for incoming frame counters needed for NWK layer security SHALL be bounded by *M*N*, where *M* and *N* represent the cardinality of *nwkSecurityMaterialSet* and *nwkNeighborTable* in the NIB, respectively.

7) If the sequence number of the received frame belongs to a newer entry in the *nwkSecurityMaterialSet*, set the *nwkActiveKeySeqNumber* to the received sequence number.

8) If there is an entry in *nwkNeighborTable* in the NIB whose extended address matches SenderAddress and whose relationship field has value 0x05 (unauthenticated child), then set relationship field in that entry to the value 0x01 (child).

4.3.2 Secured NPDU Frame

The NWK layer frame format (see section 3.3.2.2) consists of a NWK header and NWK payload field. The NWK header consists of frame control and routing fields. When security is applied to an NPDU frame, the security bit in the NWK frame control field SHALL be set to 1 to indicate the presence of the auxiliary frame header. The format for the auxiliary frame header is given in section 4.5.1. The format of a secured NWK layer frame is shown in Figure 4-3. The auxiliary frame header is situated between the NWK header and payload fields.

<table>
<tr><th>Octets: Variable</th><th>14</th><th colspan="2">Variable</th></tr>
<tr><td rowspan="2">Original NWK header ([B3], Clause 7.1)</td><td rowspan="2">Auxiliary frame header</td><td>Encrypted payload</td><td>Encrypted message integrity code (MIC)</td></tr>
<tr><td colspan="2">Secure frame payload = output of CCM</td></tr>
<tr><td colspan="2">Full NWK header</td><td colspan="2">Secured NWK payload</td></tr>
</table>

Figure 4-3. Secured NWK Layer Frame Format

4.3.3 Security-Related NIB Attributes

The NWK PIB contains attributes that are required to manage security for the NWK layer. Each of these attributes can be read and written using the NLMEGET.request and NLME-SET.request primitives, respectively. The security-related attributes contained in the NWK PIB are presented in Table 4-2, Table 4-3, and Table 4-4.

Table 4-2. NIB Security Attributes

Attribute	Identifier	Type	Range	Description	Default
nwkSecurityLevel	0xa0	Octet	0x00 – 07	The security level for outgoing and incoming NWK frames; the allowable security level identifiers are presented in Table 4-38	0x05

Attribute	Identifier	Type	Range	Description	Default
nwkSecurityMaterialSet	0xa1	A set of 2 network security material descriptors (see Table 4-3).	Variable	Set of network security material descriptors capable of maintaining an active and alternate network key.	-
nwkActiveKey SeqNumber	0xa2	Octet	0x00 – 0xFF	The sequence number of the active network key in *nwkSecurityMaterialSet.*	0x00
nwkAllFresh	0xa3	Boolean	TRUE or FALSE	Indicates whether incoming NWK frames SHALL be all checked for freshness when the memory for incoming frame counts is exceeded. See section 4.3.1.2.	TRUE

Table 4-3. Elements of the Network Security Material Descriptor

Name	Type	Range	Description	Default
KeySeqNumber	Octet	0x00 – 0xFF	A sequence number assigned to a network key by the Trust Center and used to distinguish network keys for purposes of key updates, and incoming frame security operations. This is only used when operating in a centralized security network.	00
OutgoingFrame Counter	Ordered set of 4 octets.	0x00000000 – 0xFFFFFFFF	Outgoing frame counter used for outgoing frames.	0x00000000
IncomingFrame-CounterSet	Set of incoming frame counter descriptor values. See Table 4.3.	Variable	Set of incoming frame counter values and corresponding device addresses.	Null set
Key	Ordered set of 16 octets.	-	The actual value of the key.	-

Name	Type	Range	Description	Default
NetworkKeyType	Octet	0x01 – 0x01	The type of the key. 0x01 = standard All other values are reserved.	0x01

Table 4-4. Elements of the Incoming Frame Counter Descriptor

Name	Type	Range	Description	Default
SenderAddress	Device address	Any valid 64-bit address	Extended device address.	Device-specific
IncomingFrame-Counter	Ordered set of 4 octets	0x00000000 – 0xFFFFFFFF	Incoming frame counter used for incoming frames.	0x00000000

4.3.4 Network Frame Counter Requirements

Device SHALL maintain outgoing NWK frame counters across factory resets. The outgoing NWK frame counter SHALL only be reset as detailed in this specification. A factory reset includes any over the air message, such as a NWK leave. It is permitted for manufacturers to provide a full factory reset that erases all persisted data as a separate user action.

A device can join a network, join other networks and then attempt to join the original network again. Neighbors on the original network will have a neighbor table entry for the device with the incoming frame counter set to the value that was heard when the device was previously on the network. If a fresh security material set with an outgoing NWK frame counter of zero is created when the original network is joined for a second time, devices in that network will reject frames sent with this frame counter. Devices SHALL therefore have sufficient shadow copies of their security material set and associated EPID to store the outgoing frame counter and EPID for each network that they MAY join. As an implementation optimization, it is permissible to store a single instance of the outgoing NWK frame counter that is used across all security material sets. This outgoing NWK frame counter SHALL be preserved across factory resets and when joining different networks. The only time the outgoing frame counter is reset to zero is when the device is already on a network, it receives an APSME-SWITCH-KEY and its outgoing frame counter is greater than 0x80000000.

4.3.4.1 Network Frame Counter Usage Calculations

One leap year is 366*24*60*60 = 31,622,400 seconds. The frame counter will wrap every 4,294,967,295 counts. Therefore a device would need to continuously send at a rate greater than 135 packets per second to cause the frame counter to wrap in less than a year.

Often devices do not store the exact frame counter in flash memory but use a store ahead method to prevent wearing out flash memory. This will cause the device to jump its frame counter ahead on reboot to the next higher increment. If a device increments its frame counter by 1024 on a reboot, it would have to reboot at a rate greater than once every 7 seconds to cause a wrap in a year.

A device SHALL be able to store two network keys. If there are two network key updates whilst the device is asleep or turned off, it will no longer have a valid network key and will only be able to join the network via a Trust center rejoin. Limiting the network key updates to a maximum of once every 30 days mitigates this issue.

4.4 APS Layer Security

The APS layer is responsible for the processing steps needed to securely transmit outgoing frames, securely receive incoming frames, and securely establish and manage cryptographic keys. Upper layers control the management of cryptographic keys by issuing primitives to the APS layer.

Table 4-5 lists the primitives available for key management and maintenance. Upper layers also determine which security level to use when protecting outgoing frames.

Table 4-5. The APS Layer Security Primitives

APSME Security Primitives	Request	Confirm	Indication	Response	Description
APSME-TRANSPORT-KEY	Section 4.4.2.1	-	Section 4.4.2.2	-	Transports security material from one device to another.
APSME-UPDATE-DEVICE	Section 4.4.3.1	-	Section 4.4.3.2	-	Notifies the Trust Center when a new device has joined, or an existing device has left the network.
APSME-REMOVE-DEVICE	Section 4.4.4.1	-	Section 4.4.4.2	-	Used by the Trust Center to notify a router that one of the router's child devices, or the router itself, SHOULD be removed from the network.
APSME-REQUEST-KEY	Section 4.4.5.1	-	Section 4.4.5.2	-	Used by a device to request that the Trust Center send an application link key or trust center link key.
APSME-SWITCH-KEY	Section 4.4.6.1	-	Section 4.4.6.2	-	Used by the Trust Center to tell a device to switch to a new network key.
APSME-VERIFY-KEY	Section 4.4.7.1	-	Section 4.4.7.2	-	Used by a device to verify the link key used by the trust center.
APSME-CONFIRM-KEY	Section 4.4.8.1		Section 4.4.8.2	-	Used by the trust center to confirm a previous request to verify a link key.

APSME Security Primitives	Request	Confirm	Indication	Response	Description
APSME-KEY-NEGOTIATION	Section 4.4.9.1		Section 4.4.9.2		This provides the ability for the stack to initiate or respond to a dynamic key negotiation.

4.4.1 Frame Security

The detailed steps involved in security processing of outgoing and incoming APS frames are described in sections 4.4.1.1 and 4.4.1.2, respectively.

4.4.1.1 Security Processing of Outgoing Frames

If the APS layer has a frame, consisting of a header *ApsHeader* and payload *Payload*, that needs security protection and *nwkSecurityLevel* > 0, it SHALL apply security as follows:

1) Obtain the security material and key identifier *KeyIdentifier* using the following procedure. If security material or key identifier cannot be determined, then security processing SHALL fail and no further security processing shall be done on this frame.
    a) If the frame is a result of a APSDE-DATA.request primitive:
        i) The security material associated with the destination address of the outgoing frame SHALL be obtained from the *apsDeviceKeyPairSet* attribute in the AIB. *KeyIdentifier* SHALL be set to ‘00’ (that is, a data key).
        ii) Only entries with a KeyAttribute of PROVISIONAL or VERIFIED SHALL be used. Keys with other attributes SHALL NOT be used for encryption.
    b) If the frame is a result of an APS command that requires securing.
        i) An attempt SHALL be made to retrieve the security material associated with the destination address of the outgoing frame from the *apsDeviceKeyPairSet* attribute in the AIB. Only entries with a KeyAttribute of PROVISIONAL or VERIFIED SHALL be used. Keys with other attributes SHALL NOT be used for encryption.
        ii) For all cases except transport-key commands, *KeyIdentifier* SHALL be set to ‘00’(that is, a data key). For the case of transport-key commands, *KeyIdentifier* SHALL be set to ‘02’ (that is, the key-transport key) when transporting a network key and SHALL be set to ‘03’ (that is, the key-load key) when transporting an application link key or trust center link key. See section 4.5.3 for a description of the key-transport and key-load keys.
2) Extract the outgoing frame counter (and, if KeyIdentifier is 01, the key sequence number) from the security material obtained from step 1. If the outgoing frame counter value is equal to integer 232-1, or if the key cannot be obtained, security processing SHALL fail and no further security processing shall be done on this frame.
3) Obtain the security level from the nwkSecurityLevel attribute from the NIB.
4) Construct auxiliary header AuxiliaryHeader (see section 4.5.1). The security control field SHALL be set as follows:
    a) The security level sub-field SHALL be the security level obtained from step 3.
        i) The key identifier sub-field SHALL be set to *KeyIdentifier*.
        ii) The extended nonce sub-field SHALL be set as follows: If the ApsHeader indicates the frame type is an APS Command, then the extended nonce sub-field SHALL be set to 1. Otherwise if the TxOptions bit for include extended nonce is set (0x10) then the extended nonce sub-field SHALL be set to 1. Otherwise, it SHALL be set to 0.

iii) The Frame Counter Challenge Support SHALL be set as follows: If the ApsHeader indicates the frame type is an APS Datagram, then the Require Verified Frame Counter sub-field SHALL be set to 1. Otherwise, the Require Verified Frame Counter sub-field SHALL be set to 0.

b) If the extended nonce sub-field is set to 1, then set the source address field to the 64-bit extended address of the local device.

c) The frame counter field SHALL be set to the outgoing frame counter from step 2.

d) If *KeyIdentifier* is ‘01’, the key sequence number field SHALL be present and set to the key sequence number from step 3. Otherwise, the key sequence number field SHALL NOT be present.

5) Execute the CCM mode encryption and authentication operation, as specified in section A.2, with the following exceptions:

a) The parameter *M* SHALL be obtained from Table 4-38 corresponding to the security level from step 3.

b) The bit string *Key* SHALL be the key obtained from step 1.

c) The nonce *N* SHALL be the 13-octet string constructed using the security control and frame counter fields from step 5 and the 64-bit extended address of the local device (see section 4.5.2.2).

d) If the security level requires encryption, the octet string ***a*** SHALL be the string *ApsHeader* || *AuxiliaryHeader* and the octet string *m* SHALL be the string *Payload*. Otherwise, the octet string ***a*** SHALL be the string *ApsHeader* || *AuxiliaryHeader* || *Payload* and the octet string *m* SHALL be a string of length zero.

6) If the CCM mode invoked in step 5 outputs “invalid”, security processing SHALL fail and no further security processing shall be done on this frame.

7) Let c be the output from step 5. If the security level requires encryption, the secured outgoing frame SHALL be ApsHeader || AuxiliaryHeader || c, otherwise the secured outgoing frame SHALL be ApsHeader || AuxiliaryHeader || Payload || c.

8) If the secured outgoing frame size will result in the MSDU being greater than aMaxMACFrameSize octets (see IEEE Std 802.15.4-2020 [B1]), security processing SHALL fail and no further security processing shall be done on this frame.

9) The outgoing frame counter from step 3 SHALL be incremented and stored in the appropriate location(s) of the NIB, AIB, and MAC PIB corresponding to the key that was used to protect the outgoing frame.

10) Over-write the security level sub-field of the security control field with the 3- bit all-zero string '000'.

4.4.1.2 Security Processing of Incoming Frames

If the APS layer receives a secured frame (consisting of a header *ApsHeader*, auxiliary header *AuxiliaryHeader*, and payload *SecuredPayload*) as indicated by the security sub-field of the APS header frame control field it SHALL perform security processing as follows:

1) Determine the sequence number *SequenceNumber*, key identifier *KeyIdentifier*, and received frame counter value *ReceivedFrameCounter* from the auxiliary header *AuxiliaryHeader*. If *ReceivedFrameCounter* is the 4-octet representation of the integer 2^{32}-1, security processing SHALL fail and no further security processing shall be done on this frame.

2) Determine the source address *SourceAddress* from the address-map table in the NIB, using the source address in the APS frame as the index. If the source address is incomplete or unavailable, determine if the device is joined and unauthorized. If joined and unauthorized it SHALL use the *apsDeviceKeyPairSet* that corresponds to its pre-installed link key. Otherwise, security processing SHALL fail and no further security processing shall be done on this frame.

3) Obtain the appropriate security material in the following manner. If the security material cannot be obtained, security processing SHALL fail and no further security processing shall be done on this frame.

a) If *KeyIdentifier* is ‘00’ (that is, a data key), the security material associated with the *SourceAddress* of the incoming frame SHALL be obtained from the *apsDeviceKeyPairSet* attribute in the AIB.

   b) If is ‘02’ (that is, a key-transport key), the security material associated with the *SourceAddress* of the incoming frame SHALL be obtained from the *apsDeviceKeyPairSet* attribute in the AIB; the key for this operation SHALL be derived from the security material as specified in section 4.5.3 for the key-transport key.
   c) If *KeyIdentifier* is ‘03’ (that is, a key-load key), the security material associated with the *SourceAddress* of the incoming frame SHALL be obtained from the *apsDeviceKeyPairSet* attribute in the AIB and the key for this operation SHALL be derived from the security material as specified in section 4.5.3 for the key-load key.

4) If the *apsLinkKeyType* of the associated link key is 0x00 (unique) and there is an incoming frame count *FrameCount* corresponding to *SourceAddress* from the security material obtained in step 3 and if *ReceivedFrameCount* is less than *FrameCount*, security processing SHALL fail and no further security processing shall be done on this frame.

5) Determine the security level *SecLevel* as follows. If the frame type sub-field of the frame control field of *ApsHeader* indicates an APS data frame, then *SecLevel* SHALL be set to the *nwkSecurityLevel* attribute in the NIB. Overwrite the security level sub-field of the security control field in the *AuxiliaryHeader* with the value of *SecLevel*.

6) Execute the CCM mode decryption and authentication checking operation as specified in section A.3, with the following instantiations:
   a) The parameter *M* SHALL be obtained from Table 4-38 corresponding to the security level from step 5.
      i) The bit string *Key* SHALL be the key obtained from step 3.
      ii) The nonce *N* SHALL be the 13-octet string constructed using the security control and frame counter fields from *AuxiliaryHeader*, and *SourceAddress* from step 2 (see section 4.5.2.2).
      iii) Parse the octet string *SecuredPayload* as *Payload1* || *Payload2*, where the rightmost string$Payload_2$ is an *M*-octet string. If this operation fails, security processing SHALL fail and no further security processing shall be done on this frame.
      iv) If the security level requires encryption, the octet string *a* SHALL be the string *ApsHeader* || *AuxiliaryHeader* and the octet string *c* SHALL be the string *SecuredPayload*. Otherwise, the octet string *a* SHALL be the string *ApsHeader* || *AuxiliaryHeader* || $Payload_1$ and the octet string *c* SHALL be the string $Payload_2$.

7) Return the results of the CCM operation:
   a) If the CCM mode invoked in step 6 outputs “invalid”, security processing SHALL fail and no further security processing shall be done on this frame.
   b) Let *m* be the output of step 6. If the security level requires encryption, set the octet string *UnsecuredApsFrame* to the string *ApsHeader* || *m*. Otherwise, set the octet string *UnsecuredApsFrame* to the string *ApsHeader* || *Payload*.

8) The associated apsDeviceKeyPairSet indicates support for APS Frame Counter Synchronization in the Features and Capabilities element and VerifiedFrameCounter is FALSE, security processing SHALL fail and no further security processing shall be done on this frame.
   a) The result of the failure SHALL be UNVERIFIED_FRAME_COUNTER.
   b) Otherwise security processing SHALL continue.

9) Set *FrameCount* to (*ReceivedFrameCount* + 1) and store both *FrameCount* and *SourceAddress* in the appropriate security material as obtained in step 3. If storing this frame count and address information will cause the memory allocation for this type of information to be exceeded, and the *nwkAllFresh* attribute in the NIB is TRUE, then security processing SHALL fail and no further security processing shall be done on this frame. Otherwise, security processing SHALL succeed.

4.4.1.3 Security Processing of APS Commands

A device that is not the trust center that receives an APS command SHALL determine if the message was sent by the trust center or another device for which it has a link key. If operating in a centralized security network and the message was not sent by the trust center then it SHALL discard the message and no further processing SHALL be done.

If operating in a centralized security network determining if the Trust Center sent the APS command SHALL be done as follows. If no APS encryption is present on the message then the device SHALL examine if there is an IEEE source address within the APS command frame. The IEEE source address SHALL be compared to the value of *apsTrustCenterAddress* in the AIB. If no IEEE source address is present in the APS command frame then the device SHALL verify if the NWK source of the message is 0x0000. If there is APS encryption present on the APS command then the device SHALL verify that the key used to secure the message corresponds to the *apsDeviceKeyPairSet that has a DeviceAddress equal to the value of the apsTrustCenterAddress in the AIB.*

If the message was sent by the trust center the device SHALL then consult the AIB attribute *apsLinkKeyType* associated with the sending device to determine if the key is a unique link key or Global Link key. It SHALL then consult Table 4-6 to determine the policy that shall be used.

Table 4-6. Security Policy for Accepting APS Commands in a Centralized Security Network

APS Command.	**Unique Trust Center Link Key (0x00)**	**Global Trust CenterLink Key (0x01)**
Transport Key (0x05)	APS encryption is required as per device policy (see section 4.4.1.5).	APS encryption is required as per device policy (see section 4.4.1.5).
Update Device (0x06)	APS encryption required	APS encryption not required
Remove Device (0x07)	APS encryption required	APS encryption required
Request Key (0x08)	APS encryption required Trust Center Policy MAY further restrict, see section 4.4.1.5.	APS encryption required Trust Center Policy MAY further restrict, see section 4.4.1.5.
Switch Key (0x09)	APS encryption not required	APS encryption not required
Tunnel Data (0x0E)	APS encryption not required	APS encryption not required
Verify-Key (0x0F)	APS encryption not required.	APS encryption not required
Confirm-Key (0x10)	APS encryption required	APS encryption required.

Upon reception of an APS command that does not have APS encryption but APS encryption is required by Table 4-7, the device SHALL drop the message and no further processing SHALL be done. If APS encryption is not required for the command but the received message has APS encryption, the receiving device SHALL accept and process the message. Accepting additional security on messages is required to support legacy devices in the field.

In order to support backwards compatibility with devices in the field, provisions will also be added for new devices to ensure they can interoperate with the existing devices and their legacy requirements for APS encryption.

Table 4-7. Security Policy for Sending APS Commands in a Centralized Security Network

APS Command	**Unique Trust Center Link Key**	**Global Trust Center Link Key**
Transport Key (0x05)	APS encryption MAY be optionally used. See section 4.4.1.4.	APS encryption MAY be optionally used. See section 4.4.1.4.

APS Command	Unique Trust Center Link Key	Global Trust Center Link Key
Update Device (0x06)	APS encryption SHALL be used.	APS encryption SHALL be conditionally used as per section 4.4.1.4.
Remove Device (0x07)	APS encryption SHALL be used.	APS encryption SHALL be used.
Request Key (0x08)	APS encryption SHALL be used.	APS encryption SHALL be used.
Switch Key (0x09)	APS encryption SHALL NOT be used.	APS encryption SHALL NOT be used.
Tunnel Data (0x0E)	APS encryption SHALL NOT be used.	APS encryption SHALL NOT be used.
Verify-Key (0x0F)	APS encryption SHALL NOT be used.	APS encryption SHALL NOT be used.
Confirm-Key (0x10)	APS encryption SHALL be used.	APS encryption SHALL be used.

When the local device will transmit an APS command, it shall consult Table 4-6 to determine the appropriate behavior. If APS encryption is required to be used, then the device SHALL APS encrypt the command prior to sending the message. If APS encryption is not to be used, the device SHALL NOT APS encrypt the message prior to sending the message. Conditional encryption of APS commands SHALL follow the procedure as defined by section 4.4.1.4.

4.4.1.4 Conditional Encryption of APS Commands

Devices MAY have requirements on when APS encryption SHALL or SHALL NOT be used. To ensure correct operation with those devices, the following procedure shall be undertaken as required by Table 4-6.

When sending an APS command that SHALL be conditionally encrypted, the device SHALL send the APS command with APS encryption. If the receiving device is capable of accepting APS encrypted APS commands then the sending device MAY send APS encrypted APS commands. If the receiving device is not capable of receiving APS encrypted commands, then a response to the APS command will not be received. If the receiving device is not capable of receiving APS encrypted APS commands then the sending device can either not send the APS commands or send APS commands without APS encryption.

It is left up to the implementers to determine whether or not the receiving device is capable of receiving an APS command with APS encryption. A device MAY simply send two copies of the APS command, one with APS encryption and one without, in order to satisfy the requirements of interoperability with existing devices. Note this is not for APS datagrams this is for APS Command Frames.

Conditional encryption of APS commands SHALL only apply when the *apsLinkKeyType* with receiving device is set to Global Link key (0x01).

4.4.1.5 Acceptance of Commands Based on Security Policy

There are two commands that MAY be conditionally accepted based on the local security policies in place on the device.

The APS transport key command MAY be sent with or without APS encryption. The decision to do so is based on the trust center's security policies. The trust center MAY deem it acceptable to send a key without APS encryption based on the method of transport.

Conversely, a device receiving an APS transport key command MAY choose whether or not APS encryption is required. This is most often done during initial joining. For example, during joining a device that has no preconfigured link key would only accept unencrypted transport key messages, while a device with a preconfigured link key would only accept a transport key APS encrypted with its preconfigured key.

The higher level specification implemented by the device MAY dictate the policies in place for these commands.

A device that is in the joined and authorized state SHALL accept a broadcast NWK key update sent by the Trust Center using only NWK encryption. A device that is in state of joined and unauthorized SHALL require an APS encrypted transport key if it has a preconfigured link key.

4.4.1.6 Conditional Encryption of APS Data

Devices and application profiles MAY have requirements on when APS encryption SHALL or SHALL NOT be used with normal APS Data. If the device has a set of application data encryption policies, then it SHALL encrypt any outgoing messages the policy indicates SHALL be protected. It SHALL also reject any incoming messages that are not APS encrypted when the policy indicates encryption is required.

If a device has requirements on encryption of APS data, it SHALL establish application link keys with partner devices. In a centralized security network the trust center is used to broker this link key establishment. In a distributed security network the partner devices SHALL establish a link key using an application defined method.

4.4.2 Transport-Key Services

The APSME provides services that allow an initiator to transport keying material to a responder. The different types of keying material that can be transported are shown in Table 4-9 to Table 4-12.

4.4.2.1 APSME-TRANSPORT-KEY.request

The APSME-TRANSPORT-KEY.request primitive is used for transporting a key to another device.

4.4.2.1.1 Semantics of the Service Primitive

This primitive SHALL provide the following interface:

```
APSME-TRANSPORT-KEY.request        {
                                   DestAddress,
                                   StandardKeyType,
                                   TransportKeyData,
                                   TunnelCommand
                                   TunnelCommandAddress
                                   }
```

Table 4-8 specifies the parameters for the APSME-TRANSPORT-KEY.request primitive.

Table 4-8. APSME-TRANSPORT-KEY.request Parameters

Parameter Name	Type	Valid Range	Description
DestAddress	Device address	Any valid 64-bit address	The extended 64-bit address of the destination device.
StandardKeyType	Integer	0x00 – 0x04	Identifies the type of key material that SHOULD be transported (see Table 4-9).

Parameter Name	Type	Valid Range	Description
TransportKeyData	Variable	Variable	The key being transported along with identification and usage parameters. The type of this parameter depends on the StandardKeyType parameter as follows: StandardKeyType = 0x04, Trust Center Link Key (see Table 4-10) StandardKeyType = 0x01, Standard Network Key (see Table 4-11) StandardKeyType = 0x03, Application Link Key (see Table 4-12)
TunnelCommand	Boolean	TRUE or FALSE	This indicates whether the local device SHOULD wrap the transport key message in an APS Tunnel Command. This SHOULD be done when the joining or rejoining device is not in the Trust Center's Neighbor Table (nwkNeighborTable).
TunnelCommandAddress	EUI64	Any	This indicates the destination for the APS Tunnel Command frame.

Table 4-9. StandardKeyType Parameter of the Transport-Key, Verify-Key, and Confirm-Key Primitives

Enumeration	Value	Description
Reserved	0x00	Reserved.
Standard network key	0x01	Indicates that the key is a network key to be used in standard security mode.
Reserved	0x02	Reserved.
Application link key	0x03	Indicates the key is a link key used as a basis of security between two devices.
Trust-Center link key	0x04	Indicates that the key is a link key used as a basis for security between the Trust Center and another device.
Reserved	0x05 – 0xFF	Reserved.

Table 4-10. TransportKeyData Parameter for a Trust Center Link Key

Parameter Name	Type	Valid Range	Description
Key	Set of 16 octets	Variable	The Trust Center link key.

Table 4-11. TransportKeyData Parameter for a Network Key

Parameter Name	Type	Valid Range	Description
KeySeqNumber	Octet	0x00 – 0xFF	A sequence number assigned to a network key by the Trust Center and used to distinguish network keys for purposes of key updates and incoming frame security operations.
NetworkKey	Set of 16 octets	Variable	The network key.
UseParent	Boolean	TRUE or FALSE	This parameter indicates if the destination device's parent SHALL be used to forward the key to the destination device: TRUE = Use parent FALSE = Do not use parent
ParentAddress	Device address	Any valid 64-bit address	If the UseParent is TRUE, then ParentAddress parameter SHALL contain the extended 64-bit address of the destination device's parent device; otherwise, this parameter is not used and need not be set.

Table 4-12. TransportKeyData Parameter for an Application Link Key

Parameter Name	Type	Valid Range	Description
PartnerAddress	Device address	Any valid 64-bit address	The extended 64-bit address of the device that was also sent this link key.
Initiator	Boolean	TRUE or FALSE	This parameter indicates if the destination device of this application link key requested it: TRUE = If the destination requested the key FALSE = Otherwise
Key	Set of 16 octets	Variable	The application link key

4.4.2.1.2 When Generated

The ZDO on an initiator device SHALL generate this primitive when it requires a key to be transported to a responder device.

4.4.2.1.3 Effect on Receipt

The receipt of an APSME-TRANSPORT-KEY.request primitive SHALL cause the APSME to create a transport-key command packet (see section 4.4.11.1). If the StandardKeyType parameter is 0x04 (that is, Trust Center link key), the key descriptor field of the transport-key command SHALL be set as follows:

- The key sub-field SHALL be set to the Key sub-parameter of the TransportKeyData parameter.

- The destination address sub-field SHALL be set to the DestinationAddress parameter.
- The source address sub-field SHALL be set to the local device address.

This command frame SHALL be security-protected as specified in section 4.4.1.

If the DestinationAddress parameter is all zeros, the secured command frame SHALL be unicast to any and all rx-off-when-idle children of the device. These unicasts SHALL be repeated until successful, or a subsequent APSME-TRANSPORT-KEY.request primitive with the StandardKeyType parameter equal to 0x01 has been received, or a period of twice the recommended maximum polling interval has passed.

If the StandardKeyType parameter is 0x01 (that is, a network key), the key descriptor field of the transport-key command SHALL be set as follows:

- The key sub-field SHALL be set to the Key sub-parameter of the TransportKeyData parameter.
- The sequence number sub-field SHALL be set to the KeySeqNumber sub-parameter of the TransportKeyData parameter.
- The destination address sub-field SHALL be set to the DestinationAddress parameter.
- The source address sub-field SHALL be set to the local device address.

This command frame SHALL be security-protected as specified in section 4.4.1.1 and then, if security processing succeeds, sent to the device specified by the ParentAddress sub-parameter of the TransportKeyData parameter (if the UseParent sub-parameter of the TransportKeyData parameter is TRUE) or the DestinationAddress parameter (if the UseParent sub-parameter of the TransportKeyData parameter is FALSE) by issuing a NLDE-DATA.request primitive.

If the StandardKeyType parameter is 0x03 (that is, an application link key), the key descriptor field of the transport-key command SHALL be set as follows:

- The key sub-field SHALL be set to the Key sub-parameter of the TransportKeyData parameter.
- The partner address sub-field SHALL be set to the PartnerAddress sub-parameter of the TransportKeyData parameter.
- The initiator sub-field SHALL be set 1 (if the Initiator sub-parameter of the TransportKeyData parameter is TRUE) or 0 (if the Initiator sub-parameter of the TransportKeyData parameter is FALSE).

This command frame SHALL be security-protected as specified in section 4.4.1.1 and then, if security processing succeeds, sent to the device specified by the DestinationAddress parameter by issuing a NLDE-DATA.request primitive.

If the TunnelCommand parameter is TRUE, an APS Tunnel Command SHALL be constructed as described in section 4.6.3.7. It SHALL then be sent to the device specified by the TunnelAddress parameter by issuing an NLDE-DATA.request primitive.

If the TunnelCommand parameter is FALSE it is sent to the device specified by the DestAddress parameter by issuing a NLDE-DATA.request primitive.

4.4.2.2 APSME-TRANSPORT-KEY.indication

The APSME-TRANSPORT-KEY.indication primitive is used to inform the ZDO of the receipt of keying material.

4.4.2.2.1 Semantics of the Service Primitive

This primitive SHALL provide the following interface:

```
APSME-TRANSPORT-KEY.indication      {
                                    SrcAddress,
                                    StandardKeyType,
                                    TransportKeyData
                                    LinkKeyEncryption
                                    ChangeOfTrustCenterEui64
                                    }
```

Table 4-13 specifies the parameters of the APSME-TRANSPORT-KEY.indication primitive.

Table 4-13. APSME-TRANSPORT-KEY.indication Parameters

Parameter Name	Type	Valid Range	Description
SrcAddress	Device Address	Any valid 64-bit address	The extended 64-bit address of the device that is the original source of the transported key.
StandardKeyType	Octet	0x00 – 0x06	Identifies the type of key material that was be transported; see Table 4-9.
TransportKeyData	Variable	Variable	The key being transported along with identification and usage parameters. The type of this parameter depends on the StandardKeyType parameter as follows: StandardKeyType = 0x04, Trust Center Link Key (see Table 4-10) StandardKeyType = 0x01, Standard Network Key (see Table 4-11) StandardKeyType = 0x03, Application Link Key (see Table 4-12)
LinkKeyEncryption	Boolean	TRUE or FALSE	A LinkKeyEncryption is set to TRUE if the transport key message was encrypted with a link-key, FALSE otherwise.

Parameter Name	Type	Valid Range	Description
ChangeOfTrustCenterEUI64	Boolean	TRUE or FALSE	In a centralized security network this is set to FALSE to indicate no change in the Trust Center's identity. This is the most common case. In a centralized security network this is set to TRUE when the stack has received a Transport Key from an EUI64 different than the current Trust Center. The message has valid encryption for the apsDeviceKeyPairSet entry of the current trust center but using the TrustCenterSwapOutLinkKey. This parameter is ignored in a distributed security network.

4.4.2.2.2 When Generated

The APSME SHALL generate this primitive when it receives a transport-key command as specified in section 4.4.3.3.

4.4.2.2.3 Effect on Receipt

Upon receipt of this primitive, the ZDO is informed of the receipt of the keying material.

If ChangeOfTrustCenterEUI64 has been set to true the application MAY perform additional validation of the new Trust Center application before accepting the new security material.

If ChangeOfTrustCenterEUI64 is set to true AND the application has accepted the new security material the following SHALL happen. The Application SHALL update its Trust Center Link Key after a change to the Trust Center EUI64. This MAY be done using an application defined protocol if available. Otherwise it SHALL be done using APSME-KEY-NEGOTIATION.request if the device and Trust Center Support Key Negotiation. Otherwise it MUST be done using APSME-REQUEST-KEY.request.

4.4.2.3 Upon Receipt of a Transport-Key Command

Upon receipt of a transport-key command, the APSME SHALL execute security processing as specified in, then check the key type sub-field.

The message SHALL be APS encrypted as determined by the state of the *requireLinkKeyEncryptionForApsTRansportKey* Joining Device Policy Value. If the policy is set to TRUE and the command has no APS encryption, then the message SHALL be dropped and SHALL not be processed. Otherwise, processing MAY continue.

Upon receipt of a secured transport-key command, the APSME SHALL check the key type sub-field. If the key type field is set to 0x03 (application link key) or 0x04 (Trust Center link key) and the receiving device is operating in the joined and authorized state and the command was not secured using a distributed security link key or a Trust Center link key, the command SHALL be discarded.

If the device is operating in the joined and authorized state it MAY accept a NWK broadcast transport key command with key type field set to 0x01 (standard network key) where the message has no APS encryption.

If the key type field is set to 0x01 and the command was not secured using a distributed security link key, Trust Center link key, the command SHALL be discarded.

If the key type field is set to 0x03, the APSME SHALL issue the APSME-TRANSPORT-KEY.indication primitive with the SrcAddress parameter set to the source of the key-transport command (as indicated by the NLDE-DATA.indication SrcAddress parameter), and the StandardKeyType parameter set to the key type field. The TransportKeyData parameter SHALL be set as follows:

- The Key sub-parameter SHALL be set to the key field.
- The PartnerAddress sub-parameter SHALL be set to the partner address field.
- The Initiator parameter SHALL be set to TRUE, if the initiator field is 1. Otherwise, it SHALL be set to 0.
- If the Key type field is set to 0x01, the destination field is equal to the local address, and the Source Address field is not all F's, then an APSME-TRANSPORT-KEY.indication SHALL be issued with the following parameters:
- APSME SrcAddress parameter SHALL be set to the APS Command Source Address Field.
- APSME StandardKeyType SHALL be set to 0x01.
- APSME TransportKeyData SHALL be set to the Key value of the Key Descriptor field.
- If the Key type field is set to 0x04, then an APSME-TRANSPORT-KEY.indication SHALL be issued with the following parameters:
- APSME SrcAddress parameter SHALL be set to the APS Command Source Address field.
- APSME StandardKeyType SHALL be set to 0x04.
- APSME TransportKeyData SHALL be set to the Key value of the Key Descriptor field.

If the key type field is set to 0x01 and source address field is set to 0xFFFFFFFFFFFFFFFF this indicates a distributed security network. This SHALL only be allowed if the current apsTrustCenterAddress is also 0xFFFFFFFFFFFFFFFF. Otherwise, it SHALL be dropped and no further processing shall be done. If the message is allowed, the APSME SHALL issue the APSME-TRANSPORT-KEY.indication primitive with the SrcAddress parameter set to the source address field of the key-transport command and the StandardKeyType parameter set to the key type field. The TransportKeyData parameter SHALL be set as follows: the Key subparameter SHALL be set to the key field and, in the case of a network key (that is, the key type field is set to 0x01), the KeySeqNumber sub-parameter SHALL be set to the sequence number field. The *apsTrustCenterAddress* SHOULD be set to 0xFFFFFFFFFFFFFFFF indicating a distributed security network.

If the key type field is set to 0x01 or 0x04 and the destination address field is not equal to the local address, the APSME SHALL send the command to the address indicated by the destination address field by issuing the NLDE-DATA.request primitive with security disabled.

Upon receipt of a secured transport-key command with the key type field set to 0x01, if the destination field is all zeros and the source address field is set to the value of *apsTrustCenterAddress*, the router SHALL attempt to unicast this transport-key command to any and all rx-off-when-idle children. The router SHALL continue to do so until successful, or until a subsequent transport-key command with the key type field set to 0x01 has been received, or until a period of twice the recommended maximum polling interval has passed.

4.4.3 Update Device Services

The APSME provides services that allow a device (for example, a router) to inform another device (for example, a Trust Center) that a third device has changed its status (for example, joined or left the network).APSME-UPDATE-DEVICE.request.

4.4.3.1 APSME-UPDATE-DEVICE.request

The APSME SHALL issue the APSME-UPDATE-DEVICE.request primitive when it wants to inform a device (for example, a Trust Center) that another device has a status that needs to be updated (for example, the device joined or left the network).

4.4.3.1.1 Semantics of the Service Primitive

This primitive SHALL provide the following interface:

```
APSME-UPDATE-DEVICE.request      {
                                 DestAddress,
                                 DeviceAddress,
                                 Status,
                                 DeviceShortAddress
                                 JoiningDeviceTLVs
                                 }
```

Table 4-14 specifies the parameters for the APSME-UPDATE-DEVICE.request primitive.

Table 4-14. APSME-UPDATE-DEVICE.request Parameters

Parameter Name	Type	Valid Range	Description
DestAddress	Device Address	Any valid 64-bit address	The extended 64-bit address of the device that SHALL be sent the update information.
DeviceAddress	Device Address	Any valid 64-bit address	The extended 64-bit address of the device whose status is being updated.
Status	Integer	0x00 – 0x07	Indicates the updated status of the device given by the DeviceAddress parameter: 0x00 = Standard Device Secured Rejoin 0x01 = Standard Device Unsecured Join 0x02 = Device Left 0x03 = Standard Device Trust Center Rejoin 0x04 – 0x07 = Reserved
DeviceShortAddress	Network address	0x0001 – 0xFFF7	The 16-bit network address of the device whose status is being updated.
JoiningDeviceTLVs	Octet Array	Varies	The TLVs of the joining device as relayed during Network Commissioning. Only the Joiner Encapsulation Global TLV SHALL be passed from the NLME-JOIN.indication to the APSME-UPDATE-DEVICE.request. If no TLVs are present this value SHALL be empty.

4.4.3.1.2 When Generated

The APSME (for example, on a router or Zigbee coordinator) SHALL initiate the APSME-UPDATE-DEVICE.request primitive when it wants to send updated device information to another device (for example, the Trust Center).

4.4.3.1.3 Effect on Receipt

If the local device is the Trust Center, it SHALL issue an APSME-UPDATE-DEVICE.indication, with the same parameters as the request.

If the local device is not the Trust Center, the device SHALL first create an update-device command frame (see section 4.4.9.3). The device address field of this command frame SHALL be set to the DeviceAddress parameter, the status field SHALL be set according to the Status parameter, and the device short address field SHALL be set to the DeviceShortAddress parameter. The JoinngDeviceTLVs SHALL be appended to the Update Device command frame, if present. This command frame SHALL be security-protected as specified in section 4.4.1.1 and then, if security processing succeeds, sent to the device specified in the DestAddress parameter by issuing a NLDE-DATA.request primitive.

4.4.3.2 APSME-UPDATE-DEVICE.indication

This primitive is issued to inform the APSME that it received an update-device command frame.

4.4.3.2.1 Semantics of the Service Primitive

This primitive SHALL provide the following interface:

```
APSME-UPDATE-DEVICE.indication      {
                                    SrcAddress,
                                    DeviceAddress,
                                    Status,
                                    DeviceShortAddress
                                    }
```

Table 4-15 specifies the parameters for the APSME-UPDATE-DEVICE.indication primitive.

Table 4-15. APSME-UPDATE-DEVICE.indication Parameters

Parameter Name	Type	Valid Range	Description
SrcAddress	Device Address	Any valid 64-bit address	The extended 64-bit address of the device originating the update-device command.
DeviceAddress	Device Address	Any valid 64-bit address	The extended 64-bit address of the device whose status is being updated.
Status	Integer	0x00 – 0x07	See the Status parameter definition in Table 4-14.
DeviceShortAddress	Network Address	0x0000-0xffff	The 16-bit network address of the device whose status is being updated.
JoiningDeviceTLVs	Octet Array	Varies	The TLVs of the joining device as relayed during Network Commissioning or rejoin. If no TLVs are present this value SHALL be empty.

4.4.3.2.2 When Generated

The APSME SHALL generate this primitive when it receives an update-device command frame that is successfully decrypted and authenticated, as specified in section 4.4.1.2.

4.4.3.2.3 Effect on Receipt

Upon receipt of the APSME-UPDATE-DEVICE.indication primitive, the APSME will be informed that the device referenced by the DeviceAddress parameter has undergone a status update according to the Status parameter.

If the device is not the Trust Center, this indication SHALL be ignored. No further processing shall be done.

The Trust Center SHALL process the indication as follows.

1. If the Status is Device Left, no further processing SHALL be done. A Device Left is considered informative but SHOULD NOT be considered authoritative. Security related actions SHALL not be taken on receipt of this. No further processing SHALL be done.
2. If the Trust Center's Policy indicates *allowJoins* is TRUE, it SHALL follow the procedure in 4.6.3.2.2.3.
3. Otherwise, the Trust Center SHALL follow the policy in section4.6.3.3 4.6.3.3.3.

4.4.4 Remove Device Services

The APSME provides services that allow a device (for example, a Trust Center) to inform another device (for example, a router) that one of its children SHOULD be removed from the network.

These services MAY be used in distributed network security.

4.4.4.1 APSME-REMOVE-DEVICE.request

The APSME of a device (for example, a Trust Center) SHALL issue this primitive when it wants to request that a parent device (for example, a router) remove one of its children from the network. For example, a Trust Center can use this primitive to remove a child device that is not authorized to be on the network.

4.4.4.1.1 Semantics of the Service Primitive

This primitive SHALL provide the following interface:

```
APSME-REMOVE-DEVICE.request          {
                                     ParentAddress,
                                     ChildAddress
                                     }
```

Table 4-16 specifies the parameters for the APSME-REMOVE-DEVICE.request primitive.

Table 4-16. APSME-REMOVE-DEVICE.request Parameters

Parameter Name	Type	Valid Range	Description
ParentAddress	Device Address	Any valid 64-bit address	The extended 64-bit address of the device that is the parent of the child device that is requested to be removed, or the router device that is requested to be removed.
TargetAddress	Device Address	Any valid 64-bit address	The extended 64-bit address of the target device that is requested to be removed. If a router device is requested to be removed, then the *ParentAddress* SHALL be the same as the *TargetAddress*.

4.4.4.1.2 When Generated

The APSME (for example, on a Trust Center) SHALL initiate the APSME-REMOVE-DEVICE.request primitive when it wants to request that a parent device (specified by the ParentAddress parameter) remove one of its child devices (as specified by the TargetAddress parameter), or if it wants to remove a router from the network.

If the device being removed is a router then the ParentAddress field SHALL be set to the EUI64 of that router and the TargetAddress SHALL be set to the same value.

4.4.4.1.3 Effect on Receipt

Upon receipt of the APSME-REMOVE-DEVICE.request primitive the device SHALL first create a remove-device command frame (see section 4.4.9.3). The address field of this command frame SHALL be set to the TargetAddress parameter. If the device to be removed is a router the ParentAddress and TargetAddress SHALL be the same. This command frame shall be security-protected as specified in section 4.4.1.1 and then, if security processing succeeds, sent to the device specified by the ParentAddress parameter by issuing a NLDE-DATA.request primitive.

4.4.4.2 APSME-REMOVE-DEVICE.indication

The APSME SHALL issue this primitive to inform the ZDO that it received a remove-device command frame.

4.4.4.2.1 Semantics of the Service Primitive

This primitive SHALL provide the following interface:

```
APSME-REMOVE-DEVICE.indication      {
                                    SrcAddress,
                                    ChildAddress
                                    }
```

Table 4-17 specifies the parameters for the APSME-REMOVEDEVICE.indication primitive.

Table 4-17. APSME-REMOVE-DEVICE.indication Parameters

Parameter Name	Type	Valid Range	Description
SrcAddress	Device Address	Any valid 64-bit address	The extended 64-bit address of the device requesting that a child device be removed.
TargetAddress	Device Address	Any valid 64-bit address	The extended 64-bit address of the target device that is requested to be removed.

4.4.4.2.2 When Generated

The APSME SHALL generate this primitive when it receives a remove-device command frame that is successfully decrypted and authenticated, as specified in section 4.4.1.4.

4.4.4.2.3 Effect on Receipt

Upon receipt of the APSME-REMOVE-DEVICE.indication primitive the ZDO SHALL be informed that the device referenced by the TargetAddress parameter SHALL be removed from the network.

It SHALL generate an NLME-LEAVE.request and process it as described in 3.2.2.18.

4.4.5 Request Key Services

The APSME provides services that allow a non-trust center device to request an application or trust center link key from the Trust Center. Figure 4-4 shows the processing for the request key services.

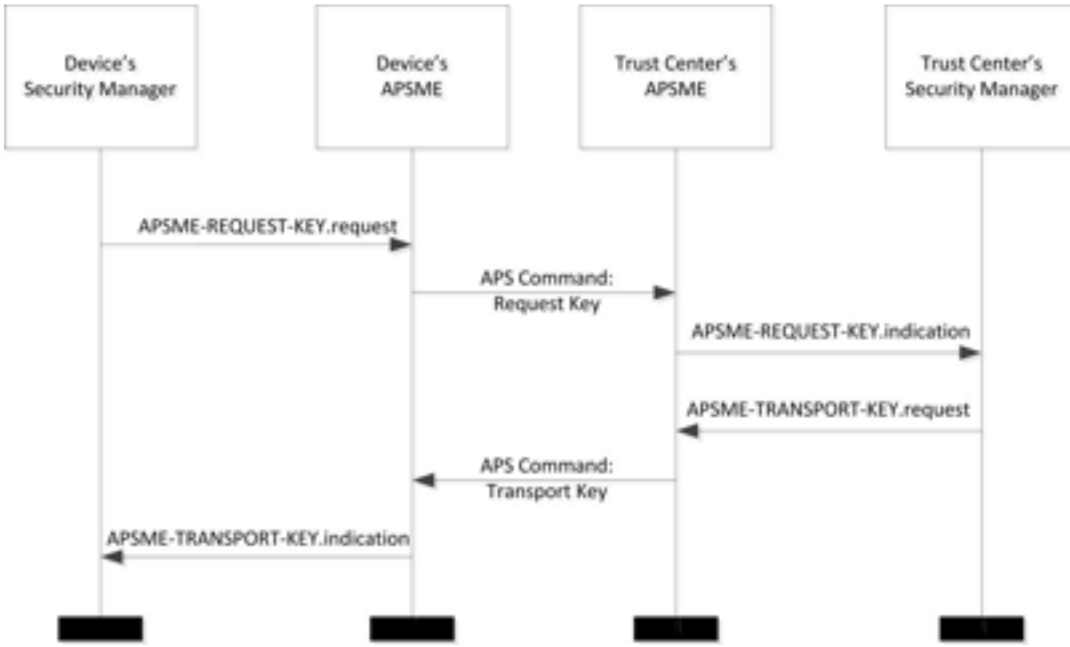


Figure 4-4. Request Key Service Processing for Trust Center Link Key

4.4.5.1 APSME-REQUEST-KEY.request

This primitive allows the Security Manager to request a new trust center link key or a new end-to-end application link key.

4.4.5.1.1 Semantics of the Service Primitive

This primitive SHALL provide the following interface:

```
APSME-REQUEST-KEY.request        {
                                 DestAddress,
                                 RequestKeyType,
                                 PartnerAddress
                                 }
```

Table 4-18 specifies the parameters for the APSME-REQUEST-KEY.request primitive.

Table 4-18. APSME-REQUEST-KEY.request Parameters

Parameter Name	Type	Valid Range	Description
DestAddress	Device Address	Any valid 64-bit address	The extended 64-bit address of the device to which the request-key command SHOULD be sent.
RequestKeyType	Octet	0x02 and 0x04	The type of key being requested. See Table 4-19.
PartnerAddress	Device Address	Any valid 64-bit address	If the RequestKeyType parameter indicates an application key, this parameter SHALL indicate an extended 64-bit address of a device that SHALL receive the same key as the device requesting the key.

Table 4-19 describes the values of the RequestKeyType enumeration. Please note that this enumeration is different than the one for the StandardKeyType in Table 4-9.

Table 4-19 RequestKeyType Values

Value	Enumeration
0x00	Reserved
0x01	Reserved
0x02	Application Link Key
0x03	Reserved
0x04	Trust Center Link Key
0x05 – 0xFF	Reserved

4.4.5.1.2 When Generated

The Security Manager of a device SHALL generate the APSME-REQUEST-KEY.request primitive when it requires either a new end-to-end application link key or trust center link key. An application link key with the Trust Center is also known as a Trust Center Link Key.

4.4.5.1.3 Effect on Receipt

Upon receipt of the APSME-REQUEST-KEY.request primitive, the device SHALL first create an APS request-key command frame (see section 4.4.9.5). The RequestKeyType field of this command frame SHALL be set to the same value as the RequestKeyType parameter. If the RequestKeyType parameter is 0x02 (that is, an application link key), then the partner address field of this command frame SHALL be the PartnerAddress parameter. Otherwise, the partner address field of this command frame SHALL NOT be present.

This command frame SHALL be security-protected as specified in section 4.4.1.1 and then, if security processing succeeds, sent to the device specified by the DestAddress parameter by issuing a NLDE-DATA.request primitive.

4.4.5.2 APSME-REQUEST-KEY.indication

The APSME SHALL issue this primitive to inform the Security Manager that it received a request-key command frame.

4.4.5.2.1 Semantics of the Service Primitive

This primitive SHALL provide the following interface:

```
APSME-REQUEST-KEY.indication    {
                                SrcAddress,
                                RequestKeyType,
                                PartnerAddress
                                }
```

Table 4-20 specifies the parameters for the APSME-REQUEST-KEY.indication primitive.

Table 4-20. APSME-REQUEST-KEY.indication Parameters

Parameter Name	Type	Valid Range	Description
SrcAddress	Device Address	Any valid 64-bit address	The extended 64-bit address of the device that sent the request-key command.
RequestKeyType	Octet	See Description.	The type of key being requested. See Table 4-19 for a list of types and valid values.
PartnerAddress	Device Address	Any valid 64-bit address	If the RequestKeyType parameter indicates an application key, this parameter SHALL indicate an extended 64-bit address of a device that SHALL receive the same key as the device requesting the key.

4.4.5.2.2 When Generated

The APSME SHALL generate this primitive when it receives a request-key command frame that is successfully decrypted and authenticated, as specified in section 4.4.1.2.

4.4.5.2.3 Effect on Receipt

Upon receipt of the APSME-REQUEST-KEY.indication primitive, the following SHALL be done:

1. If the device is not the Trust Center, the request SHALL be silently dropped and no further processing SHALL be done.
2. If the apsTrustCenterAddress of the AIB is 0xFFFFFFFFFFFFFFFF (indicating a distributed security network), the request SHALL be silently dropped and no further processing SHALL be done.
3. Find the entry in the apsDeviceKeyPairSet that the DeviceAddress matches the PartnerAddress in the interface.
    a. If no match is found, go to step 4.
    b. If a match is found and the KeyNegotiationMethod does not correspond to 0x00, APS Request Key method, then the request SHALL be dropped and no more processing SHALL be done.
4. If the RequestKeyType is 0x04, Trust Center Link Key, then follow the procedure in section 4.7.3.8.
5. If the RequestKeyType is 0x02, Application Link Key, then follow the procedure in section 4.7.3.10.
6. If the RequestKeyType is any other value, the request SHALL be silently dropped and no further processing SHALL be done.

4.4.6 Switch Key Services

The APSME provides services that allow the Trust Center to inform another device that it SHOULD switch to a new active network key.

4.4.6.1 APSME-SWITCH-KEY.request

This primitive allows a device (for example, the Trust Center) to request that another device or all devices switch to a new active network key.

4.4.6.1.1 Semantics of the Service Primitive

This primitive SHALL provide the following interface:

```
APSME-SWITCH-KEY.request        {
                                DestAddress,
                                KeySeqNumber
                                }
```

Table 4-21 specifies the parameters for the APSME-SWITCH-KEY.request primitive.

Table 4-21. APSME-SWITCH-KEY.request Parameters

Parameter Name	Type	Valid Range	Description
DestAddress	Device Address	Any valid 64-bit address	The extended 64-bit address of the device to which the switch-key command is sent. This MAY be the broadcast address 0xFFFFFFFFFFFFFFFF.
KeySeqNumber	Octet	0x00 – 0xFF	A sequence number assigned to a network key by the Trust Center and used to distinguish network keys.

4.4.6.1.2 When Generated

The ZDO of a device (for example, the Trust Center) SHALL generate the APSME-SWITCH-KEY.request primitive when it wants to inform a device or all devices to switch to a new active network key.

4.4.6.1.3 Effect on Receipt

Upon receipt of the APSME-SWITCH-KEY.request primitive, the device SHALL first create a switch-key command frame (see section 4.4.11.5). The sequence number field of this command frame SHALL be set to the same value as the KeySeqNumber parameter.

If the DestAddress is not the broadcast address 0xFFFFFFFFFFFFFFFF, this command frame SHALL be security-protected as specified in section 4.4.1.1 and then, if security processing succeeds, sent to the device specified by the DestAddress parameter by issuing a NLDE-DATA.request primitive.

If the DestAddress is the broadcast address 0xFFFFFFFFFFFFFFFF then the command SHALL NOT be security protected at the APS layer. It SHALL be sent to the NWK broadcast address 0xFFFD by issuing a NLDE-DATA.request primitive.

4.4.6.2 APSME-SWITCH-KEY.indication

The APSME SHALL issue this primitive to inform the ZDO that it received a switch-key command frame.

4.4.6.2.1 Semantics of the Service Primitive

This primitive SHALL provide the following interface:

```
APSME-SWITCH-KEY.indication     {
                                SrcAddress,
                                KeySeqNumber
                                }
```

Table 4-22 specifies the parameters for the APSME-SWITCH-KEY.indication primitive.

Table 4-22. APSME-SWITCH-KEY.indication Parameters

Parameter Name	Type	Valid Range	Description
SrcAddress	Device Address	Any valid 64-bit address	The extended 64-bit address of the device that sent the switch-key command.
KeySeqNumber	Octet	0x00 – 0xFF	A sequence number assigned to a network key by the Trust Center and used to distinguish network keys.

4.4.6.2.2 When Generated

The APSME SHALL generate this primitive when it receives a switch-key command frame that is successfully decrypted and authenticated, as specified in section 4.4.1.4.

4.4.6.2.3 Effect on Receipt

Upon receipt of the APSME-SWITCH-KEY.indication primitive the ZDO SHALL be informed that the device referenced by the SrcAddress parameter is requesting that the network key referenced by the KeySeqNumber parameter become the new active network key.

4.4.7 Verify Key Services

Figure 4-5 illustrates the flow of service requests and the over-the-air messages for the verify key.

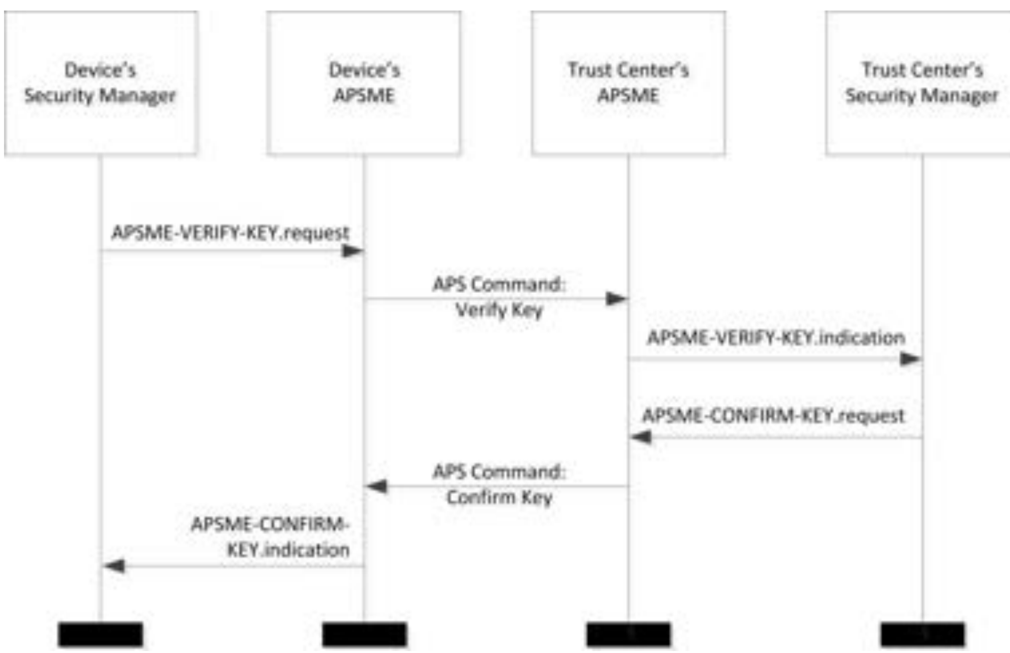


Figure 4-5. Verify-Key Processing for Trust Center Link Keys

4.4.7.1 APSME-VERIFY-KEY.request

This primitive allows a device to request that the partner device verify the Link Key between the two devices, either Trust Center Link Key or Application Link Key. When validating an application link key the frame counters are also synchronized.

4.4.7.1.1 Semantics of the Service Primitive

The primitive SHALL provide the following interface:

```
APSME-VERIFY-KEY.request        {
                                DestAddress,
                                StandardKeyType
                                }
```

Table 4-23 specifies the parameters of the APSME-VERIFY-KEY.request primitive.

Table 4-23. APSME-VERIFY-KEY.request Parameters

Parameter Name	Type	Valid Range	Description
DestAddress	Device Address	Any valid 64-bit address	The extended 64-bit address of the device to which the verify-key command be sent.
StandardKeyType	Octet	0x00 – 0xFF	Type of key being verified. See Table 4-9.

4.4.7.1.2 When Generated

The Security Manager on an initiator device SHALL generate this primitive when it wants to verify its Trust Center link key with the Trust Center.

4.4.7.1.3 Effect on Receipt

On receipt of the APSME-VERIFY-KEY.request primitive the following SHALL be performed:

1. If the local device is the Trust Center, the request is invalid and no further processing SHALL be done.
2. If the StandardKeyType parameter is not equal to 0x04 (Trust Center Link Key)and the apsTrustCenterAddress of the AIB is 0xFFFFFFFFFFFFFFFF (indicating a distributed security network), then the request is invalid. No further processing SHALL be done.
3. If the StandardKeyType parameter is equal to 0x03 (Application Link Key) and the DestAddress is equal to apsTrustCenterAddress of the AIB then the request is invalid. No further processing SHALL be done
4. The device SHALL find the corresponding entry in the apsDeviceKeyPairSet that has a DeviceAddress equal to the DestAddress of this primitive. If no entry can be found, the operation has failed and no further processing SHALL be done.
5. If the StandardKeyType is equal to 0x04 (Trust Center Link Key), then an APS Command Verify Key and APS Command Confirm Key are used to validate the Trust Center Link Key. The following additional requirements apply.
    a. The *Initiator Verify-Key Hash Value* SHALL be calculated according to section B.1.4 using the LinkKey value of the corresponding apsDeviceKeyPairSet entry found in step 5.
    b. The APSME SHALL generate an APS Command Verify-Key setting the StandardKeyType in the command to the StandardKeyType of this primitive, and setting the Hash value to the calculated Initiator Verify-Key Hash Value. The APS command SHALL NOT be APS encrypted.
6. If the StandardKeyType is equal to 0x03 (Application Link Key), then a ZDO Security_Challenge_req and ZDO Security_Challenge_rsp are used to validate the Application Link Key and synchronize frame counters.
    a. The device SHALL follow the procedure in section 4.6.3.8.1 to initiate the challenge.

4.4.7.2 APSME-VERIFY-KEY.indication

This primitive allows a device to be notified when a device is requesting to verify its Trust Center Link Key or Application Link Key. It allows the Trust Center to know when a provisional link key has been replaced by a verified link key.

4.4.7.2.1 Semantics of the Service Primitive

The primitive SHALL provide the following interface:

```
APSME-VERIFY-KEY.indication         {
                                    SrcAddress,
                                    StandardKeyType,
                                    ReceivedInitiatorHashValue
                                    ReceivedInitiatorChallengeValue
                                    }
```

Table 4-24 specifies the parameters of the APSME-VERIFY-KEY.indication primitive.

Table 4-24. APSME-VERIFY-KEY.indication Parameters

Parameter Name	Type	Valid Range	Description
SrcAddress	Device Address	Any valid 64-bit address	The extended 64-bit address of the device that sent the verify-key command.
StandardKeyType	Octet	0x00 – 0xFF	Type of key being verified. See Table 4-9.
ReceivedInitiatorHashValue	Set of 16 octets	Variable	The initiator hash of the key being verified.
ReceivedInitiatorChallengeValue	Set of 8 octets	Variable	The initiator's challenge of the key being verified.

4.4.7.2.2 When Generated

The APSME SHALL generate this primitive when it receives an APS Command Verify Key.

4.4.7.2.3 Effect on Receipt

On receipt of the APSME-VERIFY-KEY.indication primitive the following SHALL be performed:

1. If the message is a NWK broadcast, the request SHALL be dropped and no further processing SHALL be done.
2. If the StandardKeyType is 0x04 (Trust Center Link Key) and the device is not the Trust Center, or if the StandardKeyType is 0x03 (Application Link Key) and the device is operating as the Trust Center, this is not a valid request. The device SHALL follow the procedure in section 4.4.7.2.3.1 setting the Status value to 0xa3 (ILLEGAL_REQUEST). No further processing SHALL be done.
3. If the StandardKeyType parameter is not equal to 0x04 (Trust Center Link Key) and not equal to 0x03 (Application Link Key), the request is invalid. The device SHALL follow the procedure in section 4.4.7.2.3.1 setting the Status value to 0xaa (NOT_SUPPORTED). No further processing SHALL be done.

4. If the StandardKeyType is 0x04 (Trust Center Link Key) and the apsTrustCenterAddress of the AIB is set to 0xFFFFFFFFFFFFFFFF, the device is operating in a distributed security network and this is not a valid request. The device SHALL follow the procedure in section 4.4.7.2.3.1 setting the Status value to 0xaa (NOT_SUPPORTED). No further processing SHALL be done.
5. If the StantardKeyType is 0x03 (Application Link Key) and the apsTrustCenterAddress of the AIB matches the SrcAddress, then the request is invalid. The device SHALL follow the procedure in section 4.4.7.2.3.1 setting the Status value to 0xa3 (ILLEGAL_REQUEST). No further processing SHALL be done.
6. The device SHALL find the corresponding entry in the *apsDeviceKeyPairSet* attribute of the AIB where the DeviceAddress matches the SrcAddress of this primitive and the KeyAttributes is UNVERIFIED_KEY (0x01) or VERIFIED_KEY (0x02). If no entry matching those criteria is found, the following SHALL be performed.
    a. If the StandardKeyType is 0x03 (Application Link Key), then the device SHALL generate a ZDO Security_Challenge_rsp with a status of INV_REQUESTTYPE.
    b. If the StandardKeyType is 0x04 (Trust Center Link Key), the Security Manager SHALL follow the procedure in section 4.4.7.2.3.1 setting the Status value to 0xad (SECURITY_FAILURE).
    c. No further processing SHALL be done.
7. If the StandardKeyType is 0x04 (Trust Center Link Key), the device SHALL do the following:
    a. The device SHALL calculate the CalculatedInitiatorHashValue by using the LinkKey value in the corresponding *apsDeviceKeyPairSet* entry and the *Initiator Verify-Key Hash Value* cryptographic operation described in section B.1.4.
    b. The device SHALL compare the ReceivedInitiatorHashValue of the primitive with the CalculatedInitiatorHashValue. If the values do not match the operation has failed, the following SHALL be performed.
        i. The Security Manager SHALL follow the procedure in section 4.4.7.2.3.1 setting the Status value to 0xad (SECURITY_FAILURE).
        ii. No further processing SHALL be done.
8. If the StandardKeyType is 0x03 (Application Link Key), the device has already validated the re-ceived ZDO Security_Challenge_rsp as described in section .
9. The device SHALL set the KeyAttributes of the corresponding apsDeviceKeyPairSet entry to VERIFIED_KEY (0x02).
10. The device SHALL follow the procedure in section 4.4.7.2.3.1 setting the Status value to 0x00 (SUCCESS).

4.4.7.2.3.1 APSME-VERIFY-KEY.indication Response

The following shall be done when an APSME-VERIFY-KEY.indication indicates a response SHALL be generated. This procedure takes a Status code as a parameter.

An APSME-CONFIRM-KEY.request SHALL be generated with the following values:

1. The Status code SHALL be set to the Status code passed to this procedure.
2. The DestAddress SHALL be set to the SrcAddress of the APSME-VERIFY-KEY.indication.
3. The StandardKeyType SHALL be set to the StandardKeyType of the APSME-VERIFY-KEY.indication.
4. Set the Challenge value of the APSME-CONFIRM-KEY.request to the ReceivedInitiatorChallengeValue that was passed to the APSME-VERIFY-KEY.indication primitive.

4.4.8 Confirm-Key Services

4.4.8.1 APSME-CONFIRM-KEY.request

This primitive allows a Trust Center to respond to a device requesting to verify its Trust Center Link Key.

4.4.8.1.1 Semantics of the Service Primitive

The primitive SHALL provide the following interface:

```
APSME-CONFIRM-KEY.request        {
                                 Status
                                 DestAddress,
                                 StandardKeyType
                                 Challenge
                                 }
```

Table 4-25 specifies the parameters of the APSME-CONFIRM-KEY.request primitive.

Table 4-25. APSME-CONFIRM-KEY.request Parameters

Parameter Name	Type	Valid Range	Description
Status	Integer	0x00 – 0xFF	A value indicating the success or failure of a previous attempt to verify the trust center link key. See Table 2.27.
DestAddress	Device Address	Any valid 64-bit address	The extended 64-bit address of the device that sent the verify-key command.
StandardKeyType	Octet	0x00-0xFF	Type of key being verified. See Table 4-9.
Challenge	Set of 8 Octets	Varies	The challenge received for Frame Counter Verification requests.

4.4.8.1.2 When Generated

The Security Manager SHALL generate this primitive when it wants to respond to a previously received APSME-VERIFY-KEY.indication.

4.4.8.1.3 Effect on Receipt

On receipt of the APSME-CONFIRM-KEY.request primitive the following SHALL be performed:

1. If the StandardKeyType is 0x04 (Trust Center Link Key) and the device is not the Trust Center, this is not a valid request. The request SHALL be dropped and no further processing SHALL be done.
2. If the StandardKeyType parameter is not equal to 0x04 (Trust Center Link Key) and not equal to 0x03 (Application Link Key), the request is invalid. No further processing SHALL be done.
3. If the StandardKeyType is equal to 0x04 (Trust Center Link Key) and the apsTrustCenterAddress of the AIB is set to 0xFFFFFFFFFFFFFFFF, the device is operating in a distributed security network and this is not a valid request. The request SHALL be dropped and no further processing SHALL be done.
4. The device SHALL find the corresponding entry in the *apsDeviceKeyPairSet* attribute of the AIB by examining the DeviceAddress of all entries and comparing it to the DestAddress of this primitive. If no match is found, the request is invalid.
    a. The device SHALL send an APS Command Confirm Key Response to the DestAddress setting the StandardKeyType to the StandardKeyType of this primitive, the Status in the Command to FAILURE. The APS Command SHALL NOT be APS encrypted.
    b. No further processing SHALL be done.

5. If the StandardKeyType is 0x04 (Trust Center Link Key) and the device SHALL send an APS Command Confirm Key Response to the DestAddress setting the StandardKeyType to the StandardKeyType of this primitive, the Status in the Command to the Status passed to this primitive. The APS Command SHALL be APS encrypted.
6. If the StandardKeyType is 0x03 (Application Link Key) the device SHALL follow the procedure in section 4.6.3.8.5 using the Challenge received in this primitive to construct the response message.
7. The device SHALL set the IncomingFrameCounter of the apsDeviceKeyPairSet entry to 0.

4.4.8.2 APSME-CONFIRM-KEY.indication

This primitive notifies a device of the result of a previous APSME-VERIFY-KEY.request and allows it to remove a provisional link key used for joining.

4.4.8.2.1 Semantics of the Service Primitive

The primitive SHALL provide the following interface:

```
APSME-CONFIRM-KEY.indication      {
                                  Status
                                  SrcAddress,
                                  StandardKeyType,
                                  VerifiedFrameCounter,
                                  FrameCounterValue
                                  }
```

Table 4-26 specifies the parameters of the APSME-CONFIRM-KEY.indication primitive.

Table 4-26. APSME-CONFIRM-KEY.indication Parameters

Parameter Name	Type	Valid Range	Description
Status	Integer	0x00 – 0xFF	The result of the APSME-VERIFY-KEY.request operation.
SrcAddress	Device Address	Any valid 64-bit address	The extended 64-bit address of the device that sent the verify-key command.
StandardKeyType	Octet	0x00-0xFF	Type of key being verified. See Table 4-9.
VerifiedFrameCounter	Boolean	TRUE or FALSE	If TRUE, the value passed to FrameCounterValue has been verified as the latest. If FALSE, the value of FrameCounterValue SHALL be ignored.
FrameCounterValue	Integer	$0 – 2^{32}$	The frame counter that has been used in the frame counter verification.

4.4.8.2.2 When Generated

The APSME SHALL generate this primitive when it receives an APS Command Confirm Key.

4.4.8.2.3 Effect on Receipt

On receipt of the APSME-CONFIRM-KEY.indication primitive the following SHALL be performed:

1. If the message is a NWK broadcast, the request SHALL be dropped and no further processing SHALL be done.
2. If the local device is the Trust Center, this primitive is invalid. No further processing SHALL be done.
3. If the Status parameter is equal to 0x00 (Success), the operation was successful. No further processing SHALL be done.
4. If the StandardKeyType parameter is equal to 0x04 (Trust Center Link Key) and the apsTrustCenterAddress of the AIB is 0xFFFFFFFFFFFFFFFF (indicating a distributed security network), this primitive is invalid. No further processing SHALL be done.
5. If StandardKeyType is 0x04 (Trust Center Link Key) and the the SrcAddress parameter is not the equal to the apsTrustCenterAddress of the AIB, then this primitive shall be silently dropped. No further processing SHALL be done.
6. The device SHALL find the corresponding entry in the apsDeviceKeyPairSet of the AIB where the DeviceAddress is equal to the SrcAddress passed to this primitive. If no entry can be found, no further processing SHALL be done.
7. The device SHALL set the keyAttributes of the corresponding apsDeviceKeyPairSet entry to 0x02 (VERIFIED_KEY). No further processing SHALL be done.
8. If VerifiedFrameCounter is TRUE SHALL do the following:
    a. Set the IncomingFrameCounter of the corresponding apsDeviceKeyPairSet entry to FrameCounterValue.
    b. Set the VerifiedFrameCounter of the corresponding apsDeviceKeyPairSet entry to TRUE.
    c. No further processing SHALL be done.

4.4.9 Key Negotiation Services

Key Negotiation services allow the pair of devices to securely negotiate a link key using Diffie-Hellman. Each side will exchange Public Point data through the ZDO Security_Start_Key_Negotiation_req and ZDO Security_Start_Key_Negotiation_rsp and derive a link key. This is used for Dynamic Key Negotiation Joining and Dynamic Key Negotiation Update after joining.

The Security Manager on the device will determine if the Key Negotiation is allowed, whether the requisite security material is already set, and what key negotiation methods SHALL be used. In all cases both devices SHALL have an apsDeviceKeyPairSet created with a passphrase.

Partner link keys MAY be negotiated between two devices, neither of which is the trust center. This SHALL only be done after the devices have joined the network. In that case messages are not relayed through a Relaying router. It is up to the application to determine the security policies of when keys can be negotiated and whether an anonymous passphrase or authenticated passphrase SHALL be used.

A joining or rejoining device can negotiate keys with the Trust Center prior to joining. The Trust Center indicates this by sending messages relayed through the parent router. The initiator will be the joining device and the responder will be the Trust Center.

The Trust Center advertises its general key negotiation capabilities using the Supported Key Negotiation Methods Global TLV. In most cases the Trust Center advertises multiple Key Negotiation Methods in order to support devices with different cryptographic capabilities, but it MAY require certain methods for specific devices. Devices joining the network, and devices already on the network wishing to renew their Trust Center Link Key, advertise their Key Negotiation Capabilities to the Trust Center by including the Supported Key Negotiation Methods Global TLV in the Network Commissioning command frame of the Node_Desc_req ZDO command. The Trust Center selects a particular scheme out of the union that both parties, Trust Center and partner device, support, taking into account specific pre-shared secrets on record for a particular device. The Trust Center's choice of method and pre-shared secret is conveyed in a TLV included in the Security Start Key Update Request (in case the partner device is joining) or Node_Desc_rsp (in case the partner device had already joined earlier).

The negotiated, but unverified key will be kept for apsSecurityTimeOutPeriod. It is required for both sides to verify the key within that period. Prior to initiating Key Negotiation, both the joining device and the Trust Center SHALL back up the existing APS Key Pair Table entries as needed. There are many potential reasons for failure including but not limited to exceeding prescribed timeouts, missing or truncated TLVs, deviation from the prescribed sequence, Trust Center inability to support the DLK with a specific joining device, and a device reinitiating DLK while an existing sequence is still active. If Key Negotiation fails for any reason, both devices SHALL discard any generated material and SHALL ensure that their APS Key Pair Table entry is restored, if needed, so that it is identical to what it was prior to the initiation of Key Negotiation.

After negotiating a key both devices SHALL verify the key using the APSME-VERIFY-KEY.request and APSME-CONFIRM-KEY.request services.

Figure 4-6 shows how the interfaces and over the air messages are related.

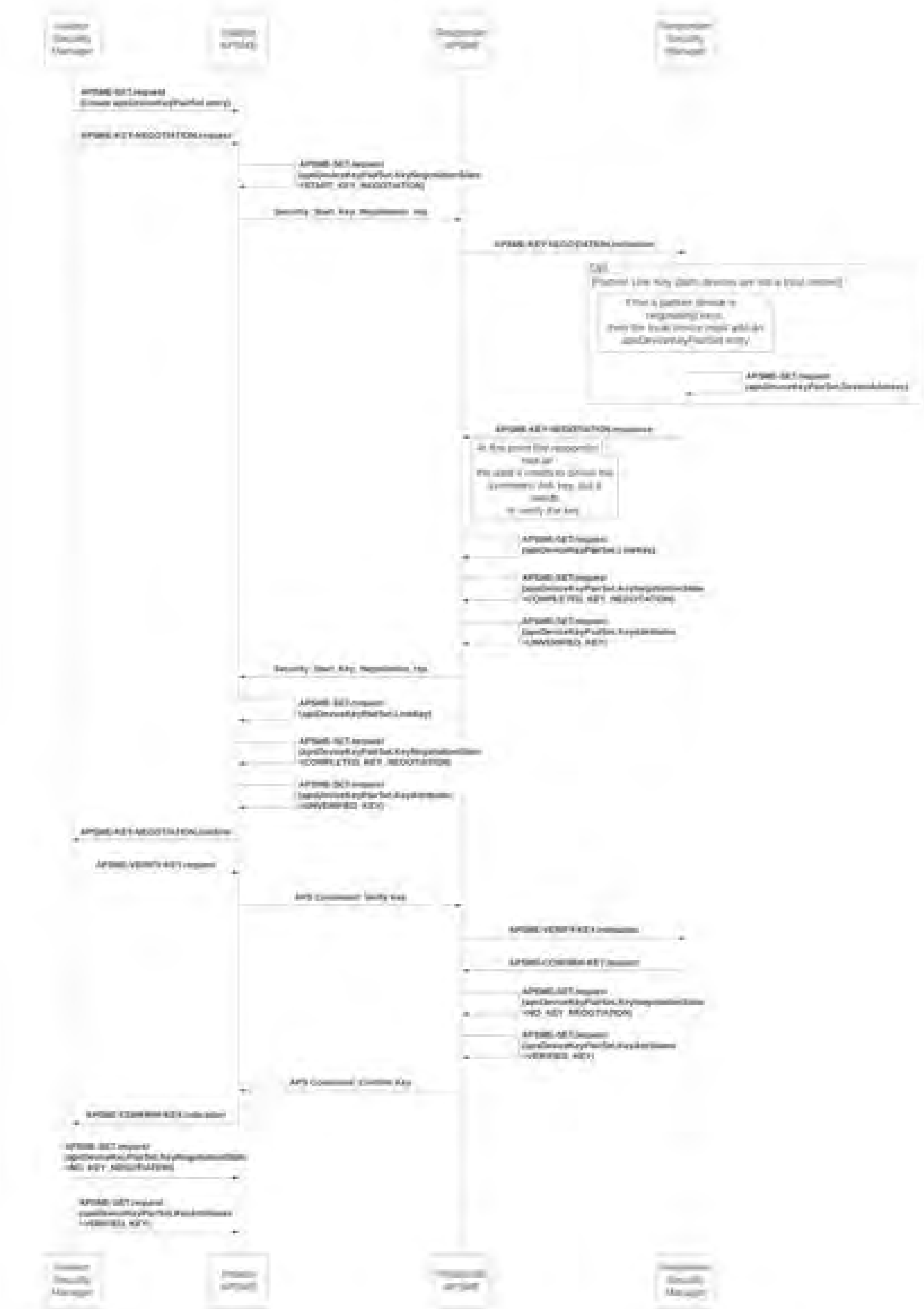


Figure 4-6. Key Negotiation Interfaces and Over the Air Message Flow

4.4.9.1 APSME-KEY-NEGOTIATION.request

This primitive requests that the APSME construct and send a ZDO Security_Start_Key_Negotiation_req frame to negotiate a new link key with another device. The target device MAY be either the Trust Center or another node in the network.

```
APSME-KEY-NEGOTIATE.request    {
                               RequestedKeyNegotiationMethod,
                               RequestedPreSharedSecretType,
                               PartnerLongAddress,
                               RelayCommand
                               RelayLongAddress
                               }
```

Table 4-27 specifies the parameters of the APSME-KEY-NEGOTIATE.request primitive.

Table 4-27. APSME-KEY-NEGOTIATE.request Parameters

Parameter Name	Type	Valid Range	Description
RequestedKeyNegotiationMethod	Enum	1 – 8	This is the enumeration indicating the key negotiation mechanism to use. This value is used for the Key Negotiation Req Selected Key Negotiation Method local TLV in the ZDO Security_Start_Key_Negotiation_req.
RequestedPreSharedSecretType	Enum	1 - 16	This is the enumeration indicating the source of preshared secret key data used for the requested negotiation. See Table 2-81.
PartnerLongAddress	EUI64	Any	This is the EUI64 of the partner that key negotiation is being performed.
RelayCommand	Boolean	TRUE or FALSE	This indicates whether or not the request SHOULD be relayed through another router. This is used when the partner is not authorized on the network yet.
RelayLongAddress	EUI64	Any	This indicates the address of the relay that SHOULD be used. When RelayCommand is FALSE this parameter is ignored.

4.4.9.1.1 When Generated

This primitive is generated when the application, on a non-Trust Center device, wants to negotiate a link key with a partner device. This could be done before the device has fully joined or rejoined the network, or it could be done after the device has joined the network.

When the Trust Center device wants to update the link key with another device it sends a ZDO Start_Update_Key_req to the device. The device will initiate the link key update locally by calling the APSME-KEY-NEGOTIATE.req and the Trust Center will act as the responder for the key negotiation exchange (starting with the APSME-KEY-NEGOTIATE.indication).

4.4.9.1.2 Effect On Receipt

The local device SHALL determine whether it supports the value in the RequestedKeyNegotiationMethod. It SHALL convert the enum into a bitmask and then compare it to the apsSupportedKeyNegotiationMethods value. If there is no corresponding match of bits then an error SHALL be returned to the next higher layer.

The stack SHALL determine whether an entry exists for the specified device in its apsDeviceKeyPairSet table of the AIB. If an entry does not exist where the DeviceAddress matches the PartnerLongAddress than an error SHALL be returned to the higher layer and no further processing SHALL be done.

The device SHALL construct a ZDO Security_Start_Key_Negotiation_req frame. The RequestedKeyNegotiationMethod SHALL be set based on the type of Public Point TLV in the Security_Start_Key_Negotiation_req (i.e. if Public Point TLV provided to Security_Start_Key_Negotiation_req is for Curve25519 then the RequestedKeyNegotiationMethod is the value for Curve25519). The device SHALL generate a public / private key pair based on the RequestedKeyNegotiationMethod. A Public Point TLV SHALL be constructed as indicated in Table 2-71 with the public point data being placed in the TLV.

For the corresponding entry in the apsDeviceKeyPairSet the device SHALL set KeyNegotiationState to START_KEY_NEGOTIATION, and the KeyNegotationMethod to the value passed as RequestedKeyNegotiationMethod.

If RelayCommand is TRUE then it SHALL construct an APS Command Relay Message Upstream. The Source EUI64 in the APS Command SHALL be the local device address, and the Message to be relayed SHALL be the ZDO Security_Start_Key_Negotiation_req. The APS Command Relay Message Upstream SHALL be sent to the RelayLongAddress.

Otherwise if RelayCommand is FALSE, the device SHALL send the ZDO Security_Start_Key_Negotiation_req to the PartnerLongAddress via an APSDE-DATA.request.

4.4.9.2 APSME-KEY-NEGOTIATION.indication

This primitive indicates that the APSME has received a request to negotiate a new link key.

```
APSME-KEY-NEGOTIATE.indication      {
                                    RequestedKeyNegotiationMethod,
                                    RequestedPreSharedSecretType,
                                    PartnerLongAddress,
                                    PublicPointData,
                                    RelayCommand,
                                    RelayLongAddress
                                    }
```

Table 4-28 specifies the parameters of the APSME-KEY-NEGOTIATE.indication primitive.

Table 4-28. APSME-KEY-NEGOTIATE.indication Parameters

Parameter Name	Type	Valid Range	Description
RequestedKeyNegotiationMethod	Enum	1 – 8	This is the enumeration indicating the key negotiation mechanism being requested.
RequestedPreSharedSecretType	Enum	1 - 16	This is the enumeration indicating the source of preshared secret key data used for the requested negotiation. See Table 2-81.
PartnerLongAddress	EUI64	Any	This is the EUI64 of the partner that is requesting the key negotiation.

Parameter Name	Type	Valid Range	Description
PublicPointData	Array	Any	This is the Public Point Data generated by the initiator for negotiating a link key.
RelayCommand	Boolean	TRUE or FALSE	This indicates whether the request was relayed through another router. This is used when the source is not authorized on the network yet.
RelayLongAddress	EUI64	Any	This indicates the address of the relay that SHOULD be used. When Relay-Command is FALSE this parameter SHALL be ignored.

4.4.9.2.1 When Generated

This is generated by the ZDO when it wants to notify the application that it has received an over-the-air request to negotiate a new key.

4.4.9.2.2 Effect On Receipt

1. Upon receipt a remote device SHALL consult the next higher layer rules for whether or not this key negotiation is allowed with the sending device. If the key negotiation is not allowed, then an APSME-KEY-NEGOTIATION.response with a ZdoStatus of NOT_AUTHORIZED SHALL be generated and no further processing SHALL be done.
2. If the device temporarily cannot handle the request due to resource constraints, then it SHALL generate an APSME-KEY-NEGOTIATION.response with a ZdoStatus of TEMPORARY_FAILURE
3. Search the apsDeviceKeyPairSet table for an entry where DeviceAddress matches the PartnerLongAddress passed to this primitive. If no entry exists then do the following
    a. Generate an APSME-KEY-NEGOTIATION.response with a ZdoStatus of NOT_AUTHORIZED.
    b. No more processing SHALL be done.
4. Determine if Key negotiation is already in progress.
    a. Examine the KeyNegotiationState in the matching apsDeviceKeyPairSet. If it is either 0x00 (None) or 0x03 (Key Negotiation Complete), go to step 4.
    b. If the KeyNegotiationState is any other value, then the device SHALL generate an APSME-KEY-NEGOTIATION.response with a ZdoStatus of SECURITY_FAIL.
5. Notify the Security Manager. The Security Manager can choose how to respond to the device and issue an APSME-KEY-NEGOTIATION.response.

4.4.9.3 APSME-KEY-NEGOTIATION.response

This primitive tells the APSME to respond to a Key negotiation request.

```
APSME-KEY-NEGOTIATE.response    {
                                Status,
                                KeyNegotiationMethod,
                                PreSharedSecretType,
                                PartnerPublicPointData,
                                PartnerLongAddress,
                                RelayCommand,
                                RelayLongAddress
                                }
```

Table 4-29 specifies the parameters of the APSME-KEY-NEGOTIATE.response primitive.

Table 4-29. APSME-KEY-NEGOTIATE.response Parameters

Parameter Name	Type	Valid Range	Description
ZdoStatus	Integer	0 – 255	The ZDP Status code to use in the response message.
PartnerPublicPointData	Array	Any	The Public point data received by the partner.
KeyNegotiationMethod	Enum	1 – 8	This is the enumeration indicating the key negotiation used in the response.
PreSharedSecretType	Enum	1 – 16	Indicates what kind of preshared key data will be used in key negotiation See Table 2-81.
PartnerLongAddress	EUI64	Any	This is the EUI64 of the partner that is requesting the key negotiation.
RelayCommand	Boolean	TRUE or FALSE	This indicates whether the response will be relayed through another router. This is used when the source is not authorized on the network yet.
RelayLongAddress	EUI64	Any	This indicates the address of the relay that SHOULD be used. When RelayCommand is FALSE this parameter SHALL be ignored.

4.4.9.3.1 When Generated

This is generated by the application to respond to a previously received APSME-KEY-NEGOTIATE.indication. The device MAY accept the request, reject the request, or respond asking for the device to use a different Key Negotiation Method.

4.4.9.3.2 Effect On Receipt

Upon receipt of this primitive, the APSME will do the following.

1. If ZdoStatus is not SUCCESS, do the following
   a. Generate a ZDO Security Start_Key_Negotiation_rsp with the status field equal to the ZdoStatus passed to this primitive.

   b. If ZdoStatus is set to BAD_KEY_NEGOTIATION_METHOD then construct a Key Establishment Selected Key Negotiation Method TLV and include the value for KeyNegotiationMethod passed to this primitive. Append the TLV to the Security_Start_Key_Negotiation_rsp.
   c. Go to step 6 (Sending the response).
2. If anonymous key exchange is used, PreSharedKeyType is 0, then the device SHALL do as follows:
   a. Find the EUI64 specified in PartnerLongAddress in the local *apsDeviceKeyPairSet* AIB value.
   b. If the device is found, then it SHALL examine the *PassphraseUpdateAllowed* value of the *apsDeviceKeyPairSet* entry in the AIB.
      i. If the value is FALSE, then a Security_Start_Key_Negotiation_rsp SHALL be generated with a status code of NOT_AUTHORIZED, go to step 6 (sending the response).
      ii. If the value is TRUE, then it SHALL set the passphrase of the *apsDeviceKeyPairSet* to *apscWellKnownPSK*.
   c. If the EUI64 is not found, a new apsDeviceKeySet entry SHALL be added for the corresponding Device EUI64 and the passphrase SHALL be set to *apscWellKnownPSK*.
3. If authenticated key exchange is specified, PreSharedKeyType is not 0, , then the local device SHALL do the following:
   a. It SHALL find the EUI64 specified in the PartnerLongAddress in its *apsDeviceKeyPairSet* AIB value. If no entry is found then processing SHALL fail and no further processing SHALL be done.
   b. It SHALL then lookup the passphrase associated with the device in the *apsDeviceKeyPairSet* AIB structure. If no passphrase is found, this is considered a failure and no further processing SHALL be done.
   c. If a failure occurs, a Security_Start_Key_Negotiation_rsp SHALL be generated with a status code of NOT_AUTHORIZED, go to step 6 (Sending the response).
   d. If no failure occurs, continue processing and go to step 4 (executing cryptographic routine).
4. Generate a local Public & Private Key pair.
5. Using previously obtained passphrase, the PartnerPublicPointData, the local public & private key pair, and the KeyNegotiationMethod the Cryptographic routine in ANNEX J SHALL be executed. If that cryptographic operation fails, then a Security_Start_Key_Negotiation_rsp SHALL be generated with a status code of SECURITY_FAIL. Go to step 6 (sending the response).
6. On success of the cryptographic operation, the device SHALL update the *apsDeviceKeyPairSet* as follows.
   a. Set LinkKey to the derived key.
   b. Set the KeyAttributes to 0x01, UNVERIFIED_KEY.
   c. Set the Timeout attribute to the value of *apsSecurityTimeOutPeriod*.
   d. Set the KeyNegotiationState to 0x02, COMPLETE_KEY_NEGOTIATION.
   e. Set the KeyNegotiationMethod to the value of KeyNegotiationMethod passed to this primitive.
   f. Set the Frame Counter Synchronization bit in the Features & Capabilities bitmap to '1'.
7. The device sends the constructed response message to the PartnerLongAddress as follows:
   a. If RelayCommand was set to TRUE.
      i. If the device is the Trust Center, it SHALL construct an APS Command Relay Message Downstream with the Destination Address set to the PartnerLong. The message to relay SHALL be the ZDO Security_Start_Key_Negotiation_rsp.
      ii. Otherwise if the device is not the Trust Center, it SHALL construct an APS Command Relay Message Upstream with the Source Address set to the PartnerLong. The message to relay SHALL be the ZDO Security_Start_Key_Negotiation_rsp.
   b. If RelayCommand was set to FALSE:
      i. The device SHALL send the ZDO Security_Start_Key_Negotiation_rsp without using a relay command frame encapsulation.

On success, both devices have an unverified, dynamically negotiated link key. It is EXPECTED that the initiator will start the verification process with APSME-VERIFY-KEY.request after the responder completes the APSME-KEY-NEGOTIATE.response.

4.4.9.4 APSME-KEY-NEGOTIATION.confirm

```
APSME-KEY-NEGOTIATE.confirm     {
                                ZdoStatus,
                                PartnerPublicPointData,
                                PartnerLongAddress,
                                RelayCommand,
                                RelayLongAddress
                                }
```

Table 4-30 specifies the parameters of the APSME-KEY-NEGOTIATE.confirm primitive.

Table 4-30. APSME-KEY-NEGOTIATE.confirm Parameters

Parameter Name	Type	Valid Range	Description
ZdoStatus	Integer	0 – 255	The ZDP Status code to use in the response message.
PartnerPublicPointData	Array	Any	The Public point data received by the partner.
PartnerLongAddress	EUI64	Any	This is the EUI64 of the partner that is requesting the key negotiation.
RelayCommand	Boolean	TRUE or FALSE	This indicates whether the response will be relayed through another router. This is used when the source is not authorized on the network yet.
RelayLongAddress	EUI64	Any	This indicates the address of the relay that SHOULD be used. When RelayCommand is FALSE this parameter SHALL be ignored.

4.4.9.4.1 When Generated

This is generated by the ZDO when it wants to notify the application that is has received an over-the-air response to negotiate a key

4.4.9.4.2 Effect On Receipt

Upon receipt of this primitive, the APSME will do the following.

1. Find the corresponding apsDeviceKeyPairSet entry that has a DeviceAddress that matches the PartnerLongAddress.
2. If no matching entry can be found, then no further processing SHALL be done.
3. If ZdoStatus is NOT success, do the following:
    a. Set the KeyNegotiationState to NO_KEY_NEGOTIATION.
    b. No further processing SHALL be done.
4. Generate a local public & private key pair.
5. Using the Passphrase in the apsDeviceKeyPairSet entry along with the PartnerPublicPointData, and the local public & private key pair, execute the cryptographic operation in ANNEX J.
6. On success of the cryptographic operation, the device SHALL update the *apsDeviceKeyPairSet* as follows.
    a. Set LinkKey to the derived key.
    b. Set the KeyAttributes to 0x01, UNVERIFIED_KEY.
    c. Set the TimeoutAttribute to the value of *apsSecurityTimeOutPeriod*.
    d. Set the KeyNegotiationState to the value of COMPLETE_KEY_NEGOTIATION.
    e. Set the KeyNegotiationMethod to the value previously passed in the APSME-KEY-NEGOTIATE.request.

On success, both devices have an unverified, dynamically negotiated link key. It is EXPECTED that the initiator will start the verification process with APSME-VERIFY-KEY.request after the responder completes the APSME-KEY-NEGOTIATE.confirm.

On failure, both devices SHALL discard any generated material and SHALL ensure that the respective APS Key Pair Table entries are identical what they were prior to initiation of Key Negotiation, as described in section 4.4.10.

4.4.10 Secured APDU Frame

The APS layer frame format consists of APS header and APS payload fields (see Figure 4-7). The APS header consists of frame control and addressing fields. When security is applied to an APDU frame, the security bit in the APS frame control field SHALL be set to 1 to indicate the presence of the auxiliary frame header. The format for the auxiliary frame header is given in section 4.5.1. The format of a secured APS layer frame is shown in Figure 4-7. The auxiliary frame header is situated between the APS header and payload fields.

<table>
<tr><th>Octets: Variable</th><th>5 or 13</th><th colspan="2">Variable</th></tr>
<tr><td rowspan="2">Original APS header
([B6], Clause 7.1)</td><td rowspan="2">Auxiliary frame header</td><td>Encrypted payload</td><td>Encrypted message integrity code (MIC)</td></tr>
<tr><td colspan="2">Secure frame payload = output of CCM</td></tr>
<tr><td colspan="2">Full APS header</td><td colspan="2">Secured APS payload</td></tr>
</table>

Figure 4-7. Secured APS Layer Frame Format

4.4.11 Command Frames

The APS layer command frame formats are given in this section.

All APS command frames SHALL set their APS frame control field as follows:

1. Set the frame type sub-field to 0x01 (Command)
2. Set the delivery-mode sub-field to 0x00 (Unicast) or 0x10 (broadcast)
3. Set the ACK format bit to 0.
4. Set the ACK request bit to 0 for APS Command Frames sent inside Tunnel Data frames from the Trust Center to a prospective joiner. A device MAY, but is not required to, set the ACK request bit to 1 for the Relay Message Upstream and Relay Message Downstream commands. A device SHALL set the ACK request bit to 1 for all other unicast APS command frames as well as command frames within the Relay Message Upstream and Relay Message Downstream commands.
5. Set the extended nonce sub field to 1 if APS security was applied. Otherwise, set it to 0.[9]
6. Set the security bit according to section 4.4.1.3 Security Processing of APS Commands.

Command identifier values are shown in Table 4-31.

[9] CCB 2432

Table 4-31. Command Identifier Values

Command Identifier	Value
Reserved	0x01
Reserved	0x02
Reserved	0x03
Reserved	0x04
APS_CMD_TRANSPORT_KEY	0x05
APS_CMD_UPDATE_DEVICE	0x06
APS_CMD_REMOVE_DEVICE	0x07
APS_CMD_REQUEST_KEY	0x08
APS_CMD_SWITCH_KEY	0x09
Reserved	0x0A
Reserved	0x0B
Reserved	0x0C
Reserved	0x0D
APS_CMD_TUNNEL	0x0E
APS_CMD_VERIFY_KEY	0x0F
APS_CMD_CONFIRM_KEY	0x10
APS_CMD_RELAY_MESSAGE_DOWNSTREAM	0x11
APS_CMD_RELAY_MESSAGE_UPSTREAM	0x12

4.4.11.1 Transport-Key Commands

The transport-key command frame shall be formatted as illustrated in Figure 4-8. The optional fields of the APS header portion of the general APS frame format SHALL NOT be present.

Octets: 1	1	1	1	Variable
Frame control	APS counter	APS command identifier	StandardKeyType	Key descriptor
APS header		Payload		

Figure 4-8. Transport-Key Command Frame

4.4.11.1.1 Command Identifier Field

The command identifier field SHALL indicate the transport-key APS command type (APS_CMD_TRANSPORT_KEY, see Table 4-31).

4.4.11.1.2 StandardKeyType Field

This field is 8 -bits in length and describes the type of key being transported. The different types of keys are enumerated in Table 4-9.

4.4.11.1.3 Key Descriptor Field

This field is variable in length and SHALL contain the actual (unprotected) value of the transported key along with any relevant identification and usage parameters. The information in this field depends on the type of key being transported (as indicated by the StandardKeyType field — see Table 4-9) and shall be set to one of the formats described in the following subsections.

4.4.11.1.3.1 Trust Center Link Key Descriptor Field

If the key type field is set to 4, the key descriptor field SHALL be formatted as shown in Figure 4-9.

Octets: 16	8	8	Varies
Key	Destination address	Source address	TLVs

Figure 4-9. Trust Center Link Key Descriptor Field in Transport-Key Command

The key sub-field SHALL contain the link key that SHOULD be used for APS encryption.

The destination address sub-field SHALL contain the address of the device which SHOULD use this link key.

The source address sub-field SHALL contain the address of the Trust Center that sent the link key.

The TLVs sub-field is optional. If present, it contains one or more TLVs as described in the section 4.4.11.1.4.

4.4.11.1.3.2 Network Key Descriptor Field

If the key type field is set to 1 this field SHALL be formatted as shown in Figure 4-10.

Octets: 16	**1**	**8**	**8**
Key	Sequence number	Destination address	Source address

Figure 4-10. Network Key Descriptor Field in Transport-Key Command

The key sub-field SHALL contain a network key.

The sequence number sub-field SHALL contain the sequence number associated with this network key.

The destination address sub-field SHALL contain the address of the device which SHOULD use this network key.

If the network key is sent to a broadcast address, the destination address subfield SHALL be set to the all-zero string and SHALL be ignored upon reception.

The source address sub-field SHALL contain the address of the device (for example, the Trust Center) which originally sent this network key.

The source address field SHALL contain 0xFFFFFFFFFFFFFFFF in a distributed security network. This indicates to the receiving device this is a distributed security network with no Trust Center.

4.4.11.1.3.3 Application Link Key Descriptor Field

If the key type field is set to 2 or 3, this field SHALL be formatted as shown in Figure 4-11.

Octets: 16	**8**	**1**	**Varies**
Key	Partner address	Initiator flag	TLVs

Figure 4-11. Application Link Key Descriptor in Transport-Key Command

The key sub-field SHALL contain a link key that is shared with the device identified in the partner address sub-field.

The partner address sub-field SHALL contain the address of the other device that was sent this link key.

The initiator flag sub-field SHALL be set to 1 if the device receiving this packet requested this key. Otherwise, this sub-field SHALL be set to 0.

The TLVs sub-field is optional. If present, it contains one or more TLVs as described in the section 4.4.11.1.4.

4.4.11.1.4 TLVs

4.4.11.1.4.1 Local TLVs

This local TLV (tag ID 0x00) indicates link-key features and the peer device's link-key capabilities as shown in Figure 4-12.

Octets
Features

Figure 4-12. Format of the Link-Key Features & Capabilities TLV

The fields of the Link-Key Features & Capabilities TLV are described in Table 4-32.

Table 4-32. Fields of the Link-Key Features & Capabilities TLV

Name	Type	Valid Range	Description
Features	map8	0x00 – 0xFF	This contains the key features bitmap as specified in Table 4-36.

4.4.11.2 Update Device Commands

The APS command frame used for device updates is specified in this section. The optional fields of the APS header portion of the general APS frame format SHALL NOT be present.

The update-device command frame SHALL be formatted as illustrated in Figure 4-13.

Octets: 1	**1**	**1**	**8**	**2**	**1**	**Varies**
Frame control	APS counter	APS command identifier	Device Address	Device short address	Status	JoinerTLVs
APS Header		Payload				

Figure 4-13. Update-Device Command Frame Format

4.4.11.2.1 Command Identifier Field

The command identifier field SHALL indicate the update-device APS command type (APS_CMD_UPDATE_DEVICE, see Table 4-31).

4.4.11.2.2 Device Address Field

The device address field SHALL be the 64-bit extended address of the device whose status is being updated.

4.4.11.2.3 Device Short Address Field

The device short address field SHALL be the 16-bit network address of the device whose status is being updated.

4.4.11.2.4 Status Field

The status field SHALL be assigned a value as described for the Status parameter in Table 4-14.

4.4.11.2.5 JoinerTLVs Field

The JoinerTLVs field MAY or MAY NOT be present. This field will be one or more TLVs received during Network Commissioning by the parent router. If the joining device or parent router has implemented a version prior to R23 then the fields will not be present. Only if both joiner and router support Revision 23 or later will the Joiner TLVs field be present.

4.4.11.3 Remove Device Commands

The APS command frame used for removing a device is specified in this section. The optional fields of the APS header portion of the general APS frame format SHALL NOT be present. The remove-device command frame shall be formatted as illustrated in Figure 4-14.

Octets: 1	1	1	8
Frame control	APS counter	APS command identifier	Target address
APS Header		Payload	

Figure 4-14. Remove-Device Command Frame Format

4.4.11.3.1 Command Identifier Field

The command identifier field SHALL indicate the remove-device APS command type (APS_CMD_REMOVE_DEVICE, see Table 4-31).

4.4.11.3.2 Target Address Field

The target address field SHALL be the 64-bit extended address of the device that is requested to be removed from the network.

4.4.11.4 Request-Key Commands

The APS command frame used by a device for requesting a key is specified in this section. The optional fields of the APS header portion of the general APS frame format SHALL NOT be present.

The request-key command frame SHALL be formatted as illustrated in Figure 4-15.

Octets: 1	1	1	1	0/8
Frame control	APS counter	APS command identifier	RequestKeyType	Partner address
APS Header		Payload		

Figure 4-15. Request-Key Command Frame Format

4.4.11.4.1 Command Identifier Field

The command identifier field SHALL indicate the request-key APS command type (APS_CMD_REQUEST_KEY, see).

4.4.11.4.2 RequestKeyType Field

The key type field SHALL be set to the key being requested. Note this Key Type is different than the StandardKeyType values used in Table 4-9 for other APS Commands or other APSME primitives. The RequestKeyType field values for the APS Command Request Key are defined in Table 4-19.

4.4.11.4.3 Partner Address Field

When the RequestKeyType field is 2 (that is, an application key), the partner address field SHALL contain the extended 64-bit address of the partner device that SHALL be sent the key. Both the partner device and the device originating the request-key command will be sent the key.

When the RequestKeyType field is 4 (that is, a trust center link key), the partner address field will not be present.

4.4.11.5 Switch-Key Commands

The APS command frame used by a device for switching a key is specified in this section. The optional fields of the APS header portion of the general APS frame format SHALL NOT be present.

The switch-key command frame SHALL be formatted as illustrated in Figure 4-16.

Octets: 1	1	1	1
Frame control	APS counter	APS command identifier	Sequence number
APS Header		Payload	

Figure 4-16. Switch-key Command Frame Format

4.4.11.5.1 Command Identifier Field

The command identifier field SHALL indicate the switch-key APS command type (APS_CMD_SWITCH_KEY, see Table 4-31).

4.4.11.5.2 Sequence Number Field

The sequence number field SHALL contain the sequence number identifying the network key to be made active.

4.4.11.6 Tunnel Command

The APS command frame used by a device for sending a command to a device that lacks the current network key is specified in this section. The optional fields of the APS header portion of the general APS frame format SHALL NOT be present. The tunnel-key command frame is sent unsecured.

The tunnel-key command frame SHALL be formatted as illustrated in Figure 4-17.

Octets:1	1	1	8	2	13	Variable	4
Frame control	APS counter	APS command identifier	Destination address	Tunneled APS header	Tunneled auxiliary frame	Tunneled command	Tunneled APS MIC
APS Header		Payload					

Figure 4-17. Tunnel Command Frame Format

4.4.11.6.1 Command Identifier Field

The command identifier field SHALL indicate the tunnel APS command type (APS_CMD_TUNNEL, see Table 4-31).

4.4.11.6.2 Destination Address

The destination address field SHALL be the 64-bit extended address of the device that is to receive the tunneled command.

4.4.11.6.3 Tunneled Auxiliary Frame Field

The tunneled auxiliary frame field shall be the auxiliary frame (see section 4.5.1) used to encrypt the tunneled command. The auxiliary frame SHALL indicate that a link key was used and SHALL include the extended nonce field.

4.4.11.6.4 Tunneled Command Field

The tunneled command field SHALL be the APS command frame to be sent to the destination.

4.4.11.7 Verify-Key Command

This APS command is used by a joining device to verify its updated link key with the peer device, such as the Trust Center.

The Verify-Key Command frame is formatted as illustrated in Figure 4-18.

Octets:1	**1**	**1**	**1**	**8**	**16**
Frame control	APS counter	APS command identifier	Standard Key Type	Source address	Initiator Verify-Key Hash Value
APS Header		APS Payload			

Figure 4-18. Verify-Key Command Frame

4.4.11.7.1 Command Identifier Field

The command identifier field SHALL indicate the verify-key request command type (APS_CMD_VERIFY_KEY, see Table 4-31).

4.4.11.7.2 StandardKeyType Field

This is the type of key being verified. See Table 4-9.

4.4.11.7.3 Source Address

This Source address field SHALL be the 64-bit extended address of the partner device that the destination shares the link key with.

4.4.11.7.4 Initiator Verify-Key Hash Value

This value is the outcome of executing the specialized keyed hash function specified in section B.1.4 using a key with the 1-octet string ‘0x03’ as the input string. The resulting value SHALL NOT be used as a key for encryption or decryption.

4.4.11.8 Confirm-Key Command

This APS command is used by a device (such as the trust center) to confirm its updated link key with the peer device.

The Confirm-Key command frame is formatted as illustrated in Figure 4-19.

Octets:1	1	1	1	1	8
Frame control	APS counter	APS command identifier	Status	StandardKeyType	Destination address
APS Header		APS Payload			

Figure 4-19. Confirm-Key Command Frame

4.4.11.8.1 Command Identifier Field

The command identifier field SHALL indicate the Confirm-Key command type (APS_CMD_VERIFY_KEY_RESPONSE, see Table 4-31).

4.4.11.8.2 Status

This will be the 1-byte status code indicating the result of the operation. See Table 2.27.

4.4.11.8.3 StandardKeyType

This is the type of key being verified. See Table 4-9.

4.4.11.8.4 Destination Address

This destination address field SHALL be the 64-bit extended address of the source device of the Verify-Key message.

4.4.11.9 Relay Message Downstream Command

This APS command is used by a Trust Center to relay a message through a parent router to a joining node as shown in Figure 4-20.

Octets: 1	1	1	Varies
Frame Control	APS Counter	APS Command Identifier	TLVs
APS Header		APS Payload	

Figure 4-20. Relay Message Downstream Command Frame

4.4.11.9.1 Command Identifier Field

The command identifier field SHALL indicate the Relay Message command type (APS_CMD_RELAY_MESSAGE_DOWNSTREAM).

4.4.11.9.2 TLVs

This field contains one or more TLVs. This command SHALL have at a minimum the Relay Message TLV.

4.4.11.9.2.1 Local TLVs

4.4.11.9.2.1.1 Relay Message TLV (ID = 0)

This local TLV (tag ID 0x00) indicates the message to be relayed and the destination of the device it is relayed to as shown n Figure 4-21.

Octets: 8	Varies
Destination EUI64	Message to be relayed

Figure 4-21. Format of the Relay Message TLV

The fields of the Relay Message TLV are defined in Table 4-33.

Table 4-33. Fields of the Relay Message TLV

Name	Type	Valid Range	Description
Destination EUI64	EUI64	0x0000000000000000 – 0xFFFFFFFFFFFFFFFF	This contains the EUI64 of the unauthorized neighbor that is the intended destination of the relayed message.
Message to be relayed	Special	Varies	This contains the single APS message, or message fragment, to be relayed from the from the Trust Center to the Joining device. The message SHALL start with the APS Header of the intended recipient.

4.4.11.10 Relay Message Upstream Command

This APS command is used by an unauthorized joining node to relay a message through a parent router to the Trust Center as shown in Figure 4-22.

Octets: 1	1	1	Varies
Frame Control	APS Counter	APS Command Identifier	TLVs
APS Header		APS Payload	

Figure 4-22. Relay Message Upstream Command Frame

4.4.11.10.1 Command Identifier Field

The command identifier field SHALL indicate the Relay Message command type (APS_CMD_RELAY_MESSAGE_UPSTREAM , see Table 4-31).

4.4.11.10.2 TLVs

This field contains one or more TLVs. This command SHALL have at a minimum the Relay Message TLV.

4.4.11.10.2.1 Local TLVs

4.4.11.10.2.1.1 Relay Message TLV (ID = 0)

This local TLV (tag ID 0x00) indicates the message to be relayed and the source of the device it is being relayed from as show in Figure 4-23.

Octets: 8	Varies
Source EUI64	Message to be relayed

Figure 4-23. Format of the Relay Message TLV

The fields of the Relay Message TLV are defined in Table 4-34.

Table 4-34. Fields of the Relay Message TLV

Name	Type	Valid Range	Description
Source EUI64	EUI64	0x0000000000000000 – 0xFFFFFFFFFFFFFFFF	This contains the EUI64 of the unauthorized neighbor that is the source of the relayed message.
Message to be relayed	Special	Varies	This contains the single APS message, or message fragment, to be relayed from the joining device to the Trust Center. The message SHALL start with the APS Header of the intended recipient.

4.4.12 Security-Related AIB Attributes

The AIB contains attributes that are required to manage security for the APS layer. Each of these attributes can be read or written using the APSME-GET.request and APSME-SET.request primitives, respectively. The security-related attributes contained in the APS PIB are presented in Table 4-35.

Table 4-35. AIB Security Attributes

Attribute	ID	Type	Range	Description	Default
apsDeviceKeyPairSet	0xaa	Set of key-pair descriptor entries. See Table 4.39.	Variable	A set of key-pair descriptors containing link keys shared with other devices.	-
apsTrustCenterAddress	0xab	Device address	Any valid 64-bit address	Identifies the address of the device's Trust Center. If this value is 0xFFFFFFFFFFFFFFFF, this means that there is no Trust Center in the network and the network	0xFFFFFFFFFFFFFFFF

Attribute	ID	Type	Range	Description	Default
				is operating in distributed security mode.	
apsSecurityTimeOutPeriod	0xac	Integer	0x0000 – 0xFFFF	The period of time a device will wait for the next expected security protocol frame (in milliseconds).	10 seconds
trustCenterPolicies	0xad	-	Variable	A set of polices encoded in the trust center on how it deals with various security events. See Table 4-42.	
apsSupportedKeyNegotiationMethods	0xaf	Bitmask	Any 32-bit value	This indicates the set of supported key negotiation methods by the local device. The set of valid values corresponds to the Supported Key Negotiation Methods Global TLV. At a minimum the device SHALL support one method, the Key Request Method.	0x01
apsChallengePeriodTimeoutSeconds	0xb0	Integer	0 – 10	The timeout in seconds for how long a challenge for an APS frame counter verification is valid.	5
apsChallengePeriodRemainingSeconds	0xb1	Integer	0 – 10	The amount of time remaining for an outstanding challenge value.	0
apsChallengeValue	0xb2	Integer	Any	The value of the last challenge that was sent.	0
apsChallengeTargetEui64	0xb3	EUI64	Any	The EUI64 of the target device that the last APS frame counter challenge was sent to.	Null
apsDeviceInterviewTimeoutPeriod	0xb4	Integer	6 – 60	The timeout duration in seconds of the period of inactivity between	12

Attribute	ID	Type	Range	Description	Default
				Device Interview Frame-before the device interview session is closed.	
apsChallengeFrame-Counter	0xb5	Integer	0x00000000-0xFFFFFFFF	A special outgoing frame counter used to generate a MIC using a nonce and key used specifically for frame counter synchronization. Note: This is a 32 bit value. See Table 2-127 which specifies that the Challenge SecurityFrameCounter is 4 octets.	0

11213

11214 **Table 4-36. Elements of the Key-Pair Descriptor**

Name	Type	Range	Description	Default
Features & Capabilities	map8	0x00, 0x01	A set of feature flags pertaining to this security material or denoting the peer's support for specific APS security features: Bit #0: Frame Counter Synchronization Support When set to '1' the peer device supports APS frame counter synchronization; else, when set to '0', the peer device does not support APS frame counter synchronization. Bits #1..#7 are reserved and SHALL be set to '0' by implementations of the current Revision of this specification and ignored when processing.	0x00
DeviceAddress	Device address	Any valid 64-bit address	Identifies the address of the entity with which this key-pair is shared.	-
KeyAttributes	Enumeration	0x00 – 0x02	This indicates attributes about the key. 0x00 = PROVISIONAL_KEY 0x01 = UNVERIFIED_KEY	-

Name	Type	Range	Description	Default
			0x02 = VERIFIED_KEY	
LinkKey	Set of 16 octets	-	The actual value of the link key.	-
OutgoingFrameCounter	Set of 4 octets	0x00000000 – 0xFFFFFFFF	Outgoing frame counter for use with this link key.	0x00000000
IncomingFrameCounter	Set of 4 octets	0x00000000 – 0xFFFFFFFF	Incoming frame counter value corresponding to *DeviceAddress.*	0x00000000
apsLinkKeyType	Enumeration	0x00 – 0x01	The type of link key in use. This will determine the security policies associated with sending and receiving APS messages. 0x00 = Unique Link Key 0x01 = Global Link Key	0x00
InitialJoinAuthentication	Enumeration	0x00 – 0x03	0x00 = NO_AUTHENTICATION 0x01 = INSTALL_CODE_KEY 0x02 = ANONYMOUS_KEY_NEGOTIATION 0x03 = KEY_NEGOTIATION_WITH_AUTHENTICATION	0x00
KeyNegotiationMethod	Enumeration	0x00 – 0x08	The value of the selected TLV sent to the device.	0x00
KeyNegotiationState	Enumeration	0x00 – 0x02	0x00 = NO_KEY_NEGOTIATION 0x01 = START_KEY_NEGOTIATION 0x02 = COMPLETE_KEY_NEGOTIATION	0x00
Passphrase	Variable size with an upper bound of 16 Octets. Refer to section 4.9.7.	Any	A value that is used by both sides during dynamic key negotiation. An unset value means this key-pair entry was not dynamically negotiated. Any other value indicates the entry was dynamically negotiated.	Unset

Name	Type	Range	Description	Default
Timeout	16-bit value	0 – 0xFFFF	The timeout, in seconds, for the specified key. When this timeout expires, the key SHALL be marked EXPIRED_KEY in the KeyAttributes and the LinkKey value SHALL not be used for encryption of messages. A value of 0xFFFF for the Timeout mean the key never expires.	0xFFFF (no expiry)
PassphraseUpdateAllowed	Boolean	TRUE or FALSE	This indicates whether the particular KeyPair passphrase MAY be updated for the device. A passphrase update is normally only allowed shortly after joining. See section 4.7.2.1.	TRUE
VerifiedFrameCounter	Boolean	TRUE or FALSE	Indicates whether the incoming frame counter value has been verified through a challenge response.	FALSE
PostJoinKeyUpdateMethod	Enumeration	0x00 – 0x04	This indicates what Link Key update method was used after the device joined the network. 0x00 = Not Updated 0x01 = Key Request Method 0x02 = Unauthenticated Key Negotiation 0x03 = Authenticated Key Negotiation 0x04 = Application Defined Certificate Based Mutual Authentication	0x00
TrustCenterSwapOut-LinkKey	Set of 16 octets	Any	The key used to indicate a Trust Center Swap-out has occurred. This key SHALL always be set to a hash of the LinkKey element. If the LinkKey is updated, then this value MUST be updated as well. See section 4.7.4.1.2.4. If the entry in the apsDeviceKeyPairSet is an application link key (where local device and the partner are not Trust Centers), implementations MAY elide this element for that entry.	-

Name	Type	Range	Description	Default
isVirtualDevice	Boolean	TRUE or FALSE	If set to TRUE, the device identified by DeviceAddress is a Zigbee Direct Virtual Device (ZVD). A Trust Center SHALL NOT send network keys to this device.	FALSE

4.4.12.1 Persistence of Security-Related AIB Values

Security Related AIB values listed below SHALL be persistently stored across reboot. Exceptions are noted below.

- apsTrustCenterAddress
- apsDeviceKeyPairSet
  - All entries SHALL be backed up.
  - The following sub-elements in each entry SHALL not be persisted
    - IncomingFrameCounter
    - Timeout
    - VerifiedFrameCounter

All other values are not required to be stored across reboots.

4.4.13 Security-Related AIB Constants

Table 4-37. Security-Related AIB Constants

Constant	Description	Value
apscWellknownPSK	A pre-shared secret that is well-known. It is used in lieu of a real pre-shared secret to allow for unauthenticated key-agreement while retaining the overall message flow and structure of an authenticated key agreement protocol like SPEKE or EC-DHE-PSK.	5a 69 67 42 65 65 41 6c 6c 69 61 6e 63 65 31 38 (hexadecimal) that is, the ASCII representation of the string “ZigBeeAlliance18”
apscJoinerTLVsUnfragmented-MaxSize	The maximum size for the JoinerTLVs passed via the NLME-JOIN.indication that are relayed in the APSME-UPDATE-DEVICE.request.	79

4.5 Common Security Elements

This section describes security-related features that are used in more than one Zigbee layer. The NWK and APS layers SHALL use the auxiliary header as specified in section 4.5.1 and the security parameters specified in section 4.5.2. The formatting of all frames and fields in this specification are depicted in the order in which they are transmitted by the NWK layer, from left to right, where the leftmost bit is transmitted first in time. Bits within each field are numbered from 0 (leftmost and least significant) to k-1 (rightmost and most significant), where the length of the field is k bits.

Fields that are longer than a single octet are sent to the next layer in the order from the octet containing the lowest numbered bits to the octet containing the highest numbered bits.

4.5.1 Auxiliary Frame Header Format

The auxiliary frame header, as illustrated by Figure 4-24, SHALL include a security control field and a frame counter field, and MAY include a sender address field and key sequence number field.

Octets: 1	**4**	**0/8**	**0/1**
Security control	Frame counter	Source address	Key sequence number

Figure 4-24. Auxiliary Frame Header Format

4.5.1.1 Security Control Field

The security control field SHALL consist of a security level, a key identifier, and an extended nonce sub-field and shall be formatted as shown in Figure 4-25.

Bit: 0-2	**3-4**	**5**	**6**	**7**
Security level	Key identifier	Extended nonce	Require Verified Frame Counter	Reserved

Figure 4-25. Security Control Field Format

4.5.1.1.1 Security Level Sub-Field

The security level identifier indicates how an outgoing frame is to be secured, how an incoming frame purportedly has been secured; it also indicates whether or not the payload is encrypted and to what extent data authenticity over the frame is provided, as reflected by the length of the message integrity code (MIC). The bit-length of the MIC MAY take the values 0, 32, 64 or 128 and determines the probability that a random guess of the MIC would be correct. The security properties of the security levels are listed in Table 4-38. Note that security level identifiers are not indicative of the relative strength of the various security levels. Also note that security levels 0 and 4 SHOULD NOT be used for frame security.

Table 4-38. Security Levels Available to the NWK, and APS Layers

Security Level Identifier	Security Level Sub-Field	Security Attributes	Data Encryption	Frame Integrity (length M of MIC, in Number of Octets)
0x00	‘000’	None	OFF	NO (M = 0)
0x01	‘001’	MIC-32	OFF	YES (M=4)
0x02	‘010’	MIC-64	OFF	YES (M=8)
0x03	‘011’	MIC-128	OFF	YES (M=16)

Security Level Identifier	Security Level Sub-Field	Security Attributes	Data Encryption	Frame Integrity (length M of MIC, in Number of Octets)
0x04	‘100’	ENC	ON	NO (M = 0)
0x05	‘101’	ENC-MIC-32	ON	YES (M=4)
0x06	‘110’	ENC-MIC-64	ON	YES (M=8)
0x07	‘111’	ENC-MIC-128	ON	YES (M=16)

4.5.1.1.2 Key Identifier Sub-Field

The key identifier sub-field consists of two bits that are used to identify the key used to protect the frame. The encoding for the key identifier sub-field SHALL be as listed in Table 4-39. Key derivation is described in section 4.5.3.

Table 4-39. Encoding of the Key Identifier Sub-Field

Key Identifier	Key Identifier Sub-Field (Figure 4-19)	Description
0x00	‘00’	A data key.
0x01	‘01’	A network key.
0x02	‘10’	A key-transport key.
0x03	‘11’	A key-load key.

4.5.1.1.3 Extended Nonce Sub-Field

The extended nonce sub-field SHALL be set to 1 if the sender address field of the auxiliary header is present. Otherwise, it SHALL be set to 0.

4.5.1.1.4 Require Verified Frame Counter

This bit indicates to the receiver that it SHALL only accept the message if the receiver has verified the frame counter of the corresponding *apsDeviceKeyPairSet*. When the bit is set, and the receiver has an unverified frame counter it SHALL drop the current received message and initiate a challenge via the ZDO Security_Challenge_req. See section 4.6.3.8 for more details.

4.5.1.2 Counter Field

The counter field is used to provide frame freshness and to prevent processing of duplicate frames.

4.5.1.3 Source Address Field

The source address field SHALL only be present when the extended nonce sub-field of the security control field is 1. When present, the source address field SHALL indicate the extended 64-bit address of the device responsible for securing the frame.

4.5.1.4 Key Sequence Number Field

The key sequence number field SHALL only be present when the key identifier subfield of the security control field is 1 (that is, a network key). When present, the key sequence number field SHALL indicate the key sequence number of the network key used to secure the frame.

4.5.2 Security Parameters

This section specifies the parameters used for the CCM security operations.

4.5.2.1 CCM Mode of Operation and Parameters

Applying security to a NWK or APS frame on a particular security level corresponds to a particular instantiation of the AES-CCM mode of operation as specified in section Figure 4-25.

The nonce SHALL be formatted as specified in section 4.5.2.2.

Table 4-38 gives the relationship between the security level sub-field of the security control field (Figure 4-25), the security level identifier, and the CCM encryption/authentication properties used for these operations.

4.5.2.2 CCM Nonce

The nonce input used for the CCM encryption and authentication transformation and for the CCM decryption and authentication checking transformation consists of data explicitly included in the frame and data that both devices can independently obtain. Figure 4-26 specifies the order and length of the subfields of the CCM nonce. The nonce's security control and frame counter fields SHALL be the same as the auxiliary header’s security control and frame counter fields (as defined in section 4.5.1) of the frame being processed. The nonce’s source address field SHALL be set to the extended 64-bit IEEE address of the device originating security protection of the frame. When the extended nonce sub-field of the auxiliary header’s security control field is 1, the extended 64-bit IEEE address of the device originating security protection of the frame SHALL correspond to the auxiliary header’s source address field (as defined in section 4.5.1) of the frame being processed.

Octets: 8	**4**	**1**
Source address	Frame counter	Security control

Figure 4-26. CCM Nonce

4.5.3 Cryptographic Key Hierarchy

The link key established between two (or more) devices is used to determine related secret keys, including data keys, key-transport keys, and key-load keys. These keys are determined as follows:

1. *Key-Transport Key*. This key is the outcome of executing the specialized keyed hash function specified in section B.1.4 under the link key with the 1-octet string ‘0x00’as the input string.
2. *Key-Load Key*. This key is the outcome of executing the specialized keyed hash function specified in section B.1.4 under the link key with the 1-octet string ‘0x02’as the input string.
3. *Data Key*. This key is equal to the link key.

All keys derived from the link key SHALL share the associated frame counters. Also, all layers of Zigbee SHALL share the active network key and associated outgoing and incoming frame counters.

4.5.4 Implementation Requirements

This section provides requirements that SHOULD be followed to ensure a secure implementation.

4.5.4.1 Random Number Generator

A Zigbee device generating random keys for distribution requires a strong method of random number generation. For example, when link keys are pre-installed (for example, in the factory), a random number MAY NOT be needed.

In all cases that require random numbers, it is critical that the random numbers are not predictable or have enough entropy, so an attacker will not be able determine them by exhaustive search. Random number generation SHALL meet the random number tests specified in FIPS 140- 2 [B15]. Methods for generation of random numbers include:

1. Base the random number on random clocks and counters within the Zigbee hardware;
2. Base the random number on random external events;
3. Seed each Zigbee device with a good random number from an external source during production. This random number can then be used as a seed to generate additional random numbers.

A combination of these methods can be used. Since the random number generation is likely integrated into the Zigbee IC, its design — and hence the ultimate viability of any encryption/security scheme — is left up to the IC manufacturers.

4.6 Functional Description

This section provides detailed descriptions of how the security services SHALL be used in a Zigbee network. A description of the security initialization responsibilities for a device starting a network is given in section 4.6.1. A brief description of the Trust Center application is given in section 4.6.2. Detailed security procedures are given in section 4.6.3.

4.6.1 Zigbee Security Initialization

The device starting a network SHALL configure the security level of the network by setting the *nwkSecurityLevel* attribute in the NIB. If the *nwkSecurityLevel* attribute is set to zero, the network will be unsecured, otherwise it will be secured.

The *key* value of the *nwkSecurityMaterialSet* attribute SHALL be set to any non-zero, random number within the range of all possible values. See section 4.5.4.1 for the requirements of random number generation. The *KeySeqNumber of* the *nwkSecurityMaterialSet* SHALL be set to 0.

If it is a centralized security network then the device SHALL configure the address of the Trust Center by setting the AIB attribute *apsTrustCenterAddress*. The device forming the network MAY also set the apsTrustCenterAddress to 0xFFFFFFFFFFFFFFFF indicating a distributed security network.

4.6.2 Trust Center Application

The Trust Center application runs on a device trusted by devices within a Zigbee network to distribute keys for the purpose of network and end-to-end application configuration management. The Trust Center SHALL configure network security policies and SHALL be used to help establish end-to-end application keys. These keys SHALL be generated at random unless a key establishment protocol is used.

4.6.2.1 Distributed Security Mode

In Distributed Security Mode, there is no unique Trust Center in the network. Keys are distributed to joining devices by routers in the network using the standard transport key commands, or by other out of band methods.

4.6.2.2 Centralized Security Mode

The centralized security mode of the Trust Center is designed for applications where a centralized security device and set of security policies is required. In this mode, the Trust Center MAY maintain a list of devices, link keys and network keys with all the devices in the network; however, it SHALL maintain a network key and controls policies of network admittance. In this mode, the *nwkAllFresh* attribute in the NIB SHALL be set to FALSE.

In Centralized networks, the Trust Center SHALL be co-located with the network Coordinator for the lifetime of the network. A Trust Center that supports Key Negotiation SHALL support all cryptographic methods for anonymous key negotiation OR all cryptographic methods for Authenticated Key Negotiation. It MAY also support both sets of cryptographic methods (authenticated and anonymous).

Each device that joins the network securely SHALL either have a Global Link key or a unique link key depending upon the application in use. It is required that the trust center have prior knowledge of the value of the link key and the type (Global or unique) in order to securely join the device to the network. A Global Link key has the advantage that the memory required by the Trust Center does not grow with the number of devices in the network. A unique link key has the advantage of being unique for each device on the network and application communications can be secured from other devices on the network. Both types of keys MAY be used on the network, but a device SHALL only have one type in use per device-key pair. A joining device that supports Key Negotiation SHALL support at least one cryptographic method for anonymous Key Negotiation and one cryptographic method for Authenticated Key Negotiation. This is in addition to the mandatory Request Key Behavior.

The security policy settings for centralized security are further detailed in section 4.7.1.

4.6.3 Security Procedures

This section gives message sequence charts for joining a secured network, authenticating a newly joined device, updating the network key, recovering the network key, establishing end-to-end application keys, and leaving a secured network.

4.6.3.1 Joining a Secured Network

Figure 4-27 shows the high level flow and interfaces that are activated for a device joining.

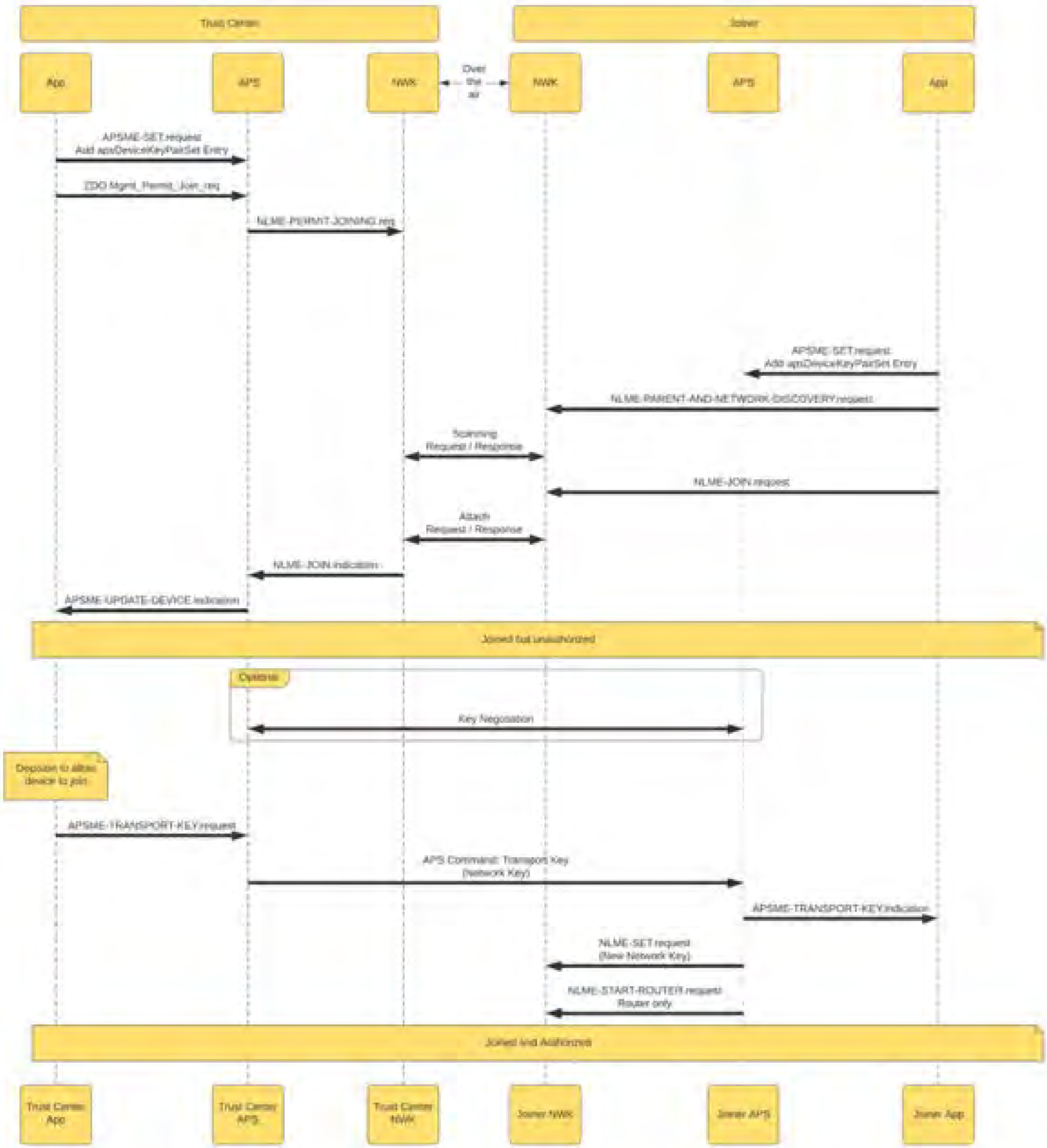


Figure 4-27. Joining a Secured Network

When a device prepares to join a secured network it SHALL create an apsDeviceKeyPairSet entry for the Trust Center with its initial joining link key. It will set the apsLinkKeyType of that entry according to the kind of key it has. If it is using the default trust center link key, or another Global Link key, it SHALL set apsLinkKeyType to 0x01. If it is using a unique link key it SHALL set apsLinkKeyType to 0x00. If it supports key negotiation it will also set its initial Passphrase attribute of the apsDeviceKeyPairSet entry.

The joiner will use the NLME-NETWORK-AND-PARENT-DISCOVERY primitives to discover a set of candidate networks and parents to join, and then use the NLME-JOIN primitives to attempt to join each one until it is accepted in the network or until all candidates are exhausted. The details of this are discussed in chapter 3 (section 3.6.1.5 and section 3.6.1.6).

After receiving a MAC Association Response or Network Commissioning Response command, the joiner device will be considered joined but unauthorized. The parent router informs the trust center of the new device (centralized network) and the Trust Center decides the next steps. If the Trust Center wants to deny the device it can send an APS Remove Device command to the parent router, or simply let the joining device timeout. If the Trust Center wants to allow the device onto the network the next steps will depend on various factors.

If both Trust Center and joiner support Dynamic Link Key Negotiation, and the intermediate router is an R23 device that can relay key negotiation messages, then the Trust Center and joiner will negotiate a link key. If key negotiation is not supported, or there is a pre-R23 device that the joiner has joined to, then the Trust Center can simply send the joiner an APS Transport key.

If a link key is negotiated prior to joining, the Trust Center will APS encrypt the APS Transport key with the new link key. Otherwise the Trust Center will encrypt the message with the previously configured link key for the device.

When joining for the first time the address of the Trust Center will not be known to the joiner. The DeviceAddress value for the apsDeviceKeyPairSet entry for the Trust Center will initially have an all F's address, as will the apsTrustCenterAddress AIB value. Once the joiner device joins the network it will learn the address of the Trust Center from the Source Address field of the APS Transport Key command. If that source address is all F's then the joiner device will know the network is a Distributed Security network. Otherwise it will know the address of the Trust Center that is operating in a Centralized Security network. It will update the apsDeviceKeyPairSet entry and apsTrustCenterAddress AIB values accordingly. Many of the security policies for the network will vary based on whether it is a distributed or centralized security network.

The Trust Center MAY require that a specific link key is used for joining to authenticate the joining device. The device MAY not know the preferences of the Trust Center and could require multiple attempts before successfully authenticating.

The Trust Center decides whether the device is allowed to join anonymously, or whether authentication is required. If anonymous joining is allowed then the key negotiation will use a well-known shared secret, or if using APS Transport Key without key negotiation then a well-known key is used to encrypt the message.

If authentication is required the Trust Center will use a static secret key or passcode known by both the joining device and trust center. This secret is relayed out of band. Authenticated key negotiation will utilize the shared secret in the cryptographic exchange to derive a link key. If just APS Transport Key is used, then the command will be encrypted with the secret key.

4.6.3.2 Authorization

Once a device joins a secured network and is declared "joined but unauthorized", it SHALL be authorized by receiving an APS transport key command containing the active network key.

4.6.3.2.1 Router Operation

If the *apsTrustCenterAddress* is 0xFFFFFFFFFFFFFFFF, this indicates a Distributed Security network. If the apsTrustCenterAddress is any other value, it indicates a Centralized Security network.

In centralized security networks, the router SHALL do the following upon receipt of an NLME-JOIN.indication.

1. If the Method parameter indicates NWK Association it SHALL do the following:
   a. Verify that JoinerTLVs are not greater than *capsJoinerTLVsUnfragmentedMaxSize* .
      i. If they are, then a NLME-JOIN.response SHALL be generated with the following parameters:
         1. STATUS is set to INV_REQUESTTYPE
         2. NetworkAddress is set to the NetworkAddress in the NLME-JOIN.indication.
         3. ExtendedAddress is set to the ExtendedAddress in the NLME-JOIN.indication.
      ii. No further processing SHALL be done on the router.
   b. If the router NLME has verified it has the resources to allow the device to join or rejoin, it SHALL do the following:
      i. Send back an NLME-JOIN.response with
         1. STATUS of SUCCESS
         2. NetworkAddress is set to the NetworkAddress in the NLME-JOIN.indication.
         3. ExtendedAddress is set to the ExtendedAddress in the NLME-JOIN.indication.

   c. Continue processing.
2. Issue an APSME-UPDATE-DEVICE.request with the following parameters:
   a. Status set to the DeviceStatus value returned via the NLME-JOIN.indication primitive.
   b. DestAddress is the apsTrustCenterAddress of the AIB
   c. DeviceAddress is set to the ExtendedAddress of the NLME-JOIN.request primitive.
   d. DeviceShortAddress is the NetworkAddress of the NLME-JOIN.request primitive.

In a Distributed Security Network no Update Device message is generated. The router SHALL issue an APSME-TRANSPORT-KEY.request with the StandardKeyType set to 0x01 (Standard Network Key) and the key value from the nwkSecurityMaterialSet of the NIB with a KeySeqNumber equal to the nwkActiveSeqNumber of the NIB. The message SHALL be APS encrypted with the Distributed Security Global Key in the apsDeviceKeyPairSet.

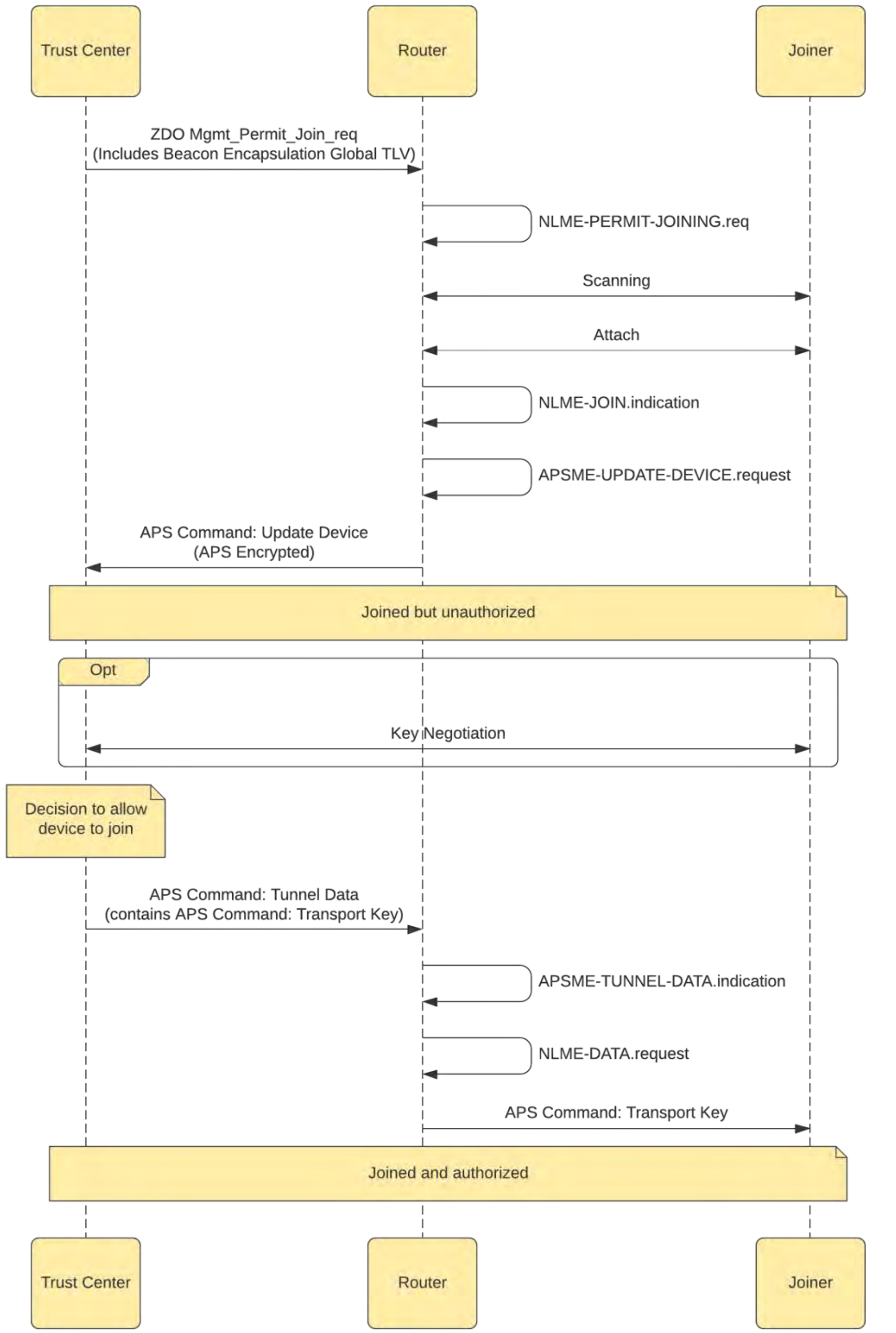
If the router is not the Trust Center, it generates an APSME-UPDATE-DEVICE.request that generates an APS Command Update Device message over-the-air to the Trust Center. If the router is the Trust Center, it generates an APSME-UPDATE-DEVICE.indication and SHALL begin the authorization procedure by simply operating as a Trust Center.

The router SHALL NOT forward messages to a child device, or respond to ZDO requests or NWK command requests on that child's behalf, while the value of the relationship field entry in the corresponding *nwkNeighborTable* in the NIB is 0x05 (unauthenticated child). It SHALL react to APS Commands Tunnel Data, and Relay Message Downstream from the Trust Center to send messages to the unauthenticated child. It SHALL also react to the APS Command Relay Message Upstream sent from the Joiner to the Router. If the relationship of the nwkNeighborTable for that child device changes or the entry is removed, the router SHALL silently reject those commands.

The APS Command Update Device, APS Command Tunnel Data, and APS Commands Relay Message Upstream/Downstream communicated between the Trust Center and the router SHALL be secured at the NWK layer by the active network key. The conditions for APS encryption of the the APS Command Update Device is described below. The transport-key command and Relay Message Downstream sent from the router to the joiner SHALL NOT be secured at the network layer.

Two copies of the update-device APS command SHALL be generated by the parent router if the apsDeviceKeyPairSet entry for the TC indicates the apsLinkKeyType is 0x01 (Global). One copy SHALL be encrypted at both the APS and the NWK layer, while the other copy SHALL only be encrypted at the NWK layer. This is done due to an interoperability issue where previously certified Trust Centers MAY have requirements on the encryption that it accepts for the APS Command Update Device message.

A device with apsDeviceKeyPairSet that has an apsLinkKeyType of 0x00 (Unique Link Key) does not have to generate two update device messages and SHALL only generate a single APS encrypted APS Command Update Device.



11457

11458 **Figure 4-28. Router Message Passing when a Device Joins the Network**

11459

4.6.3.2.2 Trust Center Operation

The Trust Center role in the authorization procedure SHALL be activated upon receipt of an APSME-UPDATE-DEVICE.indication primitive. The Trust Center behaves differently depending on the following factors:

a) Whether the Trust Center decides to allow any device to perform a first time join (for example, the Trust Center is in a mode that allows new devices to join)

b) If the Trust Center Policies require prior knowledge of the device to allow joining

If, at any time during the authorization procedure, the Trust Center decides not to allow the new device to join the network (for example, a policy decision or a failed higher level key-establishment protocol), it SHALL take actions to remove the device from the network. If the Trust Center is not the router of the newly joined device, it MAY remove the device from the network by issuing the APSME-REMOVE-DEVICE. request primitive with the ParentAddress parameter set to the address of the router originating the update-device command and the ChildAddress parameter set to the address of the joined (but unauthorized) device. Alternatively the Trust Center MAY let an unauthorized device just timeout; in that case the Trust Center will not send a removal message.

4.6.3.2.2.1 Applying Security Policies

After being activated for the authorization procedure, the Trust Center SHALL determine whether or not to allow the device onto the network. This decision will be based on its own security policies, see section 4.7.1. The Trust Center MAY also require that the device update its link key prior to joining and receiving the network key.

If the Trust Center requires key negotiation first, it SHALL follow the procedure in section 4.6.3.2.2.2. Otherwise it MAY skip to section 4.6.3.2.2.3.

4.6.3.2.2.2 Dynamic Key Negotiation Joining

If both the Trust Center and joiner support negotiation of a link key before joining, the Trust Center MAY choose to do so. This is known as Dynamic Key Negotiation Joining. The Trust Center does this by issuing a Security_Start_Key_Update_req to the joining device. The message SHALL include the following:

Selected Key Negotiation Method TLV

Prior to successful negotiation of a link key the Trust Center SHALL restrict what messages it accepts from the joiner to the following:

- ZDO Security_Start_Key_Negotiation_req
- ZDO Security_Key_Update_rsp
- APS Command: Relay Message Upstream
- APS Command: APS Confirm Key
- APS Acknowledgement frames

It then follows the procedure described in section 4.6.3.5.

If the Dynamic Key Negotiation Joining fails for any reason, both the Trust Center and the joiner SHALL discard any generated material and SHALL ensure that the respective APS Key Pair Table entries are identical with what they were prior to initiation of Key Negotiation, as described in section 4.4.9.

After successfully negotiating a link key the Trust Center MAY send the device additional application layer messages before accepting the device on the network. The messages sent are up to the higher layer application. It is recommended that the Trust Center apply restrictions to the messages it accepts from the joining device based on the messages it will generate to the device. For example, if only a ZDO Node Descriptor Request is sent then the Trust Center during joining it, then it SHOULD only accept a ZDO Node Descriptor Response. No other messages SHOULD be accepted in that example.

4.6.3.2.2.3 Initial Network Key Distribution

If the Trust Center decides to allow the device onto the network, it SHALL send the device the active network key by issuing the APSME-TRANSPORT-KEY.request primitive with the DestAddress parameter set to the address of the newly joined device, and the StandardKeyType parameter set to 0x01 (that is, standard network key).

The KeySeqNumber sub-parameter of the APSME-TRANSPORT-KEY.request SHALL be set to the sequence count value for the active network key and the NetworkKey sub-parameter SHALL be set to the active network key. The UseParent sub-parameter SHALL be set to FALSE if the Trust Center is the router; otherwise, the UseParent sub-parameter SHALL be set to TRUE and the ParentAddress sub-parameter SHALL be set to the address of the router originating the update-device command.

4.6.3.2.2.4 Managing Network Admittance of Zigbee Direct Virtual Devices

If the JoiningDeviceTLVs parameter of the APSME-UPDATE-DEVICE.indication primitive contains the Device Capability Extension Global TLV (Table I.4.8-4-64 Global TLV Definitions in Annex I) and the “Zigbee Direct Virtual Device” flag (bit #0) in this TLV indicates the joiner is a Zigbee Direct Virtual Device (ZVD), the Trust Center SHALL NOT send a transport key message with the active network key to the virtual joiner. The joiner is indicating that it does NOT need the active network key to participate in the mesh and can use a variant of the network key instead. This is further enforced by the Zigbee Direct Device (ZDD), which will not generate an update device message for a ZVD that did not include the Device Capability Extension Global TLV with the Zigbee Direct Virtual Device flag set to '1'.

Instead, if the *allowVirtualDevices* Trust Center Policy Value equals TRUE, the Trust Center SHALL construct a Transport Key message including the Basic Authorization Key for the joiner and send it to the joining virtual device via its parent router (ZDD). The Basic Authorization Key SHALL be derived as specified in section 6.3.2.1 Authorization Keys of [B2]. The Trust Center SHALL send this Transport Key message encrypted with the Trust Center Link Key for the joining ZVD and, more specifically, it SHALL apply the ‘key-load key’ derivative key as APS encryption key as opposed to the ‘key-transport key’. The ‘key-transport key’ SHALL exclusively be used for the delivery of active or prospective network keys to IEEE Std 802.15.4 Zigbee devices; the ‘key-load key’ SHALL exclusively be used for the delivery of keys to the ZVD. The Trust Center SHALL record that the joining device is a Zigbee Direct Virtual Device in the corresponding apsDeviceKeyPairSet entry for the joiner, by setting the *isVirtualDevice* element of the Key-Pair Descriptor to TRUE. It SHALL always refer back to this information when it performs a network key rotation and SHALL NOT send a prospective network key to the ZVD. Instead, the Trust Center SHALL apply the above mentioned approach to deliver an updated Basic Authorization key derived under the prospective network key to a particular Zigbee Direct Virtual Device in its *apsKeyPairSet*.

If the *allowVirtualDevices* Trust Center Policy Value equals FALSE, the Trust Center SHALL NOT admit the joiner to the network. It MAY instigate an APSME-REMOVE-DEVICE.request to the effect of notifying the Zigbee Direct Device (ZDD) acting as parent router of the ZVD of its decision to not admit the joining ZVD to the network such that the ZDD could purge the unauthenticated ZVD from its neighbor table sooner.

4.6.3.2.3 Joining Device Operation

The joining device SHALL be preconfigured with a Trust Center link key and start a timer. It sets the SecurityTimer value of the *nwkNeighborTable* entry of NIB for its parent to *apsSecurityTimeOutPeriod*. It will then wait to receive one of the following:

1) APS Command of Transport Key containing the active network key encrypted with its preconfigured link key
2) ZDO Security_Start_Key_Update_req Command indicating the device SHALL start key negotiation.

If the timer reaches zero before receiving one of the above messages, the joiner SHALL considered the join operation to have failed.

4.6.3.2.3.1 Dynamic Key Negotiation Joining

If the Trust Center and joiner both support dynamic link key negotiation, the Trust Center MAY choose to negotiate a link key with the device prior to it receiving the Network Key.

Prior to negotiating a link key, a joined but unauthorized device SHALL restrict what messages it processes to only the key negotiation messages. These include the following:

- ZDO Security_Start_Key_Update_Response
- ZDO Start Key Negotiation Response
- APS Command: Verify Key
- APS Command: Verify Key
- APS Acknowledgement frames

The Joining Device SHALL follow the procedure in section 4.6.3.2.2.2.

Once a link key has been successfully negotiated, the joiner MAY receive additional application layer messages before the Trust Center transmits a copy of the current network key. The joiner SHALL respond to those messages but MAY limit the responses based on its own security policy values. It SHALL reset the SecurityTimer value of the *nwkNeighborTable* entry of its parent in the NIB to *apsSecurityTimeOutPeriod* for every received and APS encrypted message.

Receiving a Network Key. Upon receipt of the APSME-TRANSPORT-KEY.indication primitive with the StandardKeyType parameter set to 0x01 (that is, the standard network key), the joining device SHALL set the *apsTrustCenterAddress* attribute in its AIB to the SrcAddress parameter of the APSME-TRANSPORT-KEY.indication primitive. The joining device is now considered authorized and SHALL enter the normal operating state for standard security mode.

If the *apsTrustCenterAddress* is set to 0xFFFFFFFFFFFFFFFF the network is in distributed security mode. The device SHALL enter the normal operating state.

Additional application layer security authentication or initialization MAY be required by the higher layer specification.

If the joining device did not receive any APS encrypted messages within the *apsSecurityTimeOutPeriod* since receiving the NLME-JOIN.confirm primitive, it SHALL reset and MAY choose to start the joining procedure again.

If the Dynamic Key Negotiation Joining fails for any reason, both the Trust Center and the joiner SHALL discard any generated material and SHALL ensure that the respective APS Key Pair Table entries are identical with what they were prior to initiation of Key Negotiation, as described in section 4.4.9.

4.6.3.2.3.2 Joining Complete

After successfully joining or rejoining a secured network by receiving the network key, the joining device SHALL set the *nwkSecurityLevel* attribute in the NIB to the values indicated by the stack profile.

A joined and authorized device SHALL always apply NWK layer security to outgoing frames unless the frame is destined for a newly joined but unauthorized child.

In a secured network, if the device does not become authorized within a preconfigured amount of time, it SHALL leave the network (see section 4.6.3.6.3).

4.6.3.3 Rejoining Security

Devices SHALL follow the procedures described in this section as necessary to support rejoining, in conjunction with the mechanism described in section 2.5.4.5.2.2.

A device does not have to verify its trust center link key with the APSME-VERIFY-KEY services after a rejoin.

4.6.3.3.1 Secure Rejoin

When a device is rejoining and secures the NWK rejoin request command with the active network key, no further authorization is required beyond validation of the NWK security. Both centralized and distributed networks MAY use Secure Rejoin.

Figure 4-29 shows the flow of messages during a secure rejoin. Note that in Distributed network security the APS Command Update Device SHALL NOT be sent.

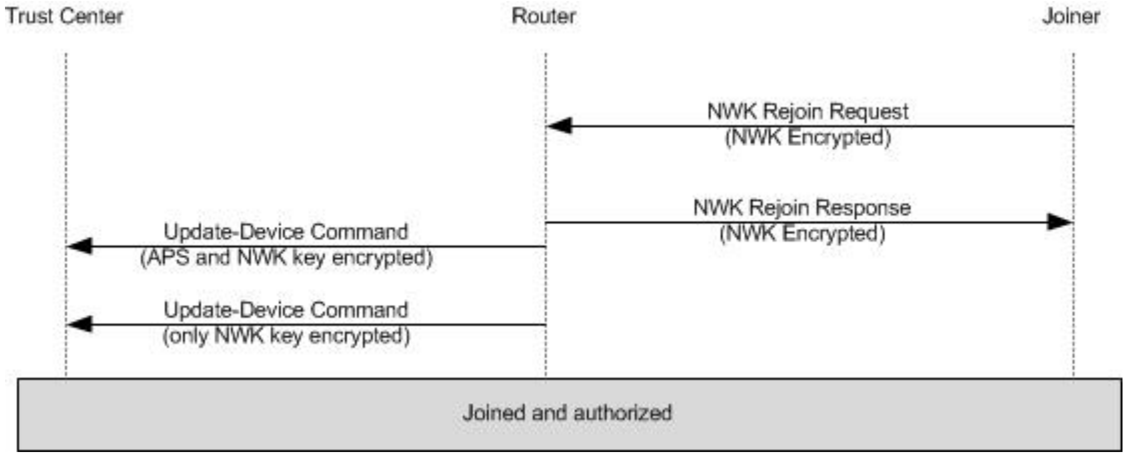


Figure 4-29. Secure Rejoin

4.6.3.3.2 Trust Center Rejoin

A Trust Center Rejoin is used when a device MAY no longer have the current network key and therefore SHOULD NOT secure the NWK rejoin command. If the network is using a different network key then the device using the old network key will be rejected. A Trust Center rejoin is a NWK Rejoin command where the command is sent without NWK layer security and allows a device to request the current active network key.

Figure 4-30 illustrates a trust center rejoin operation.

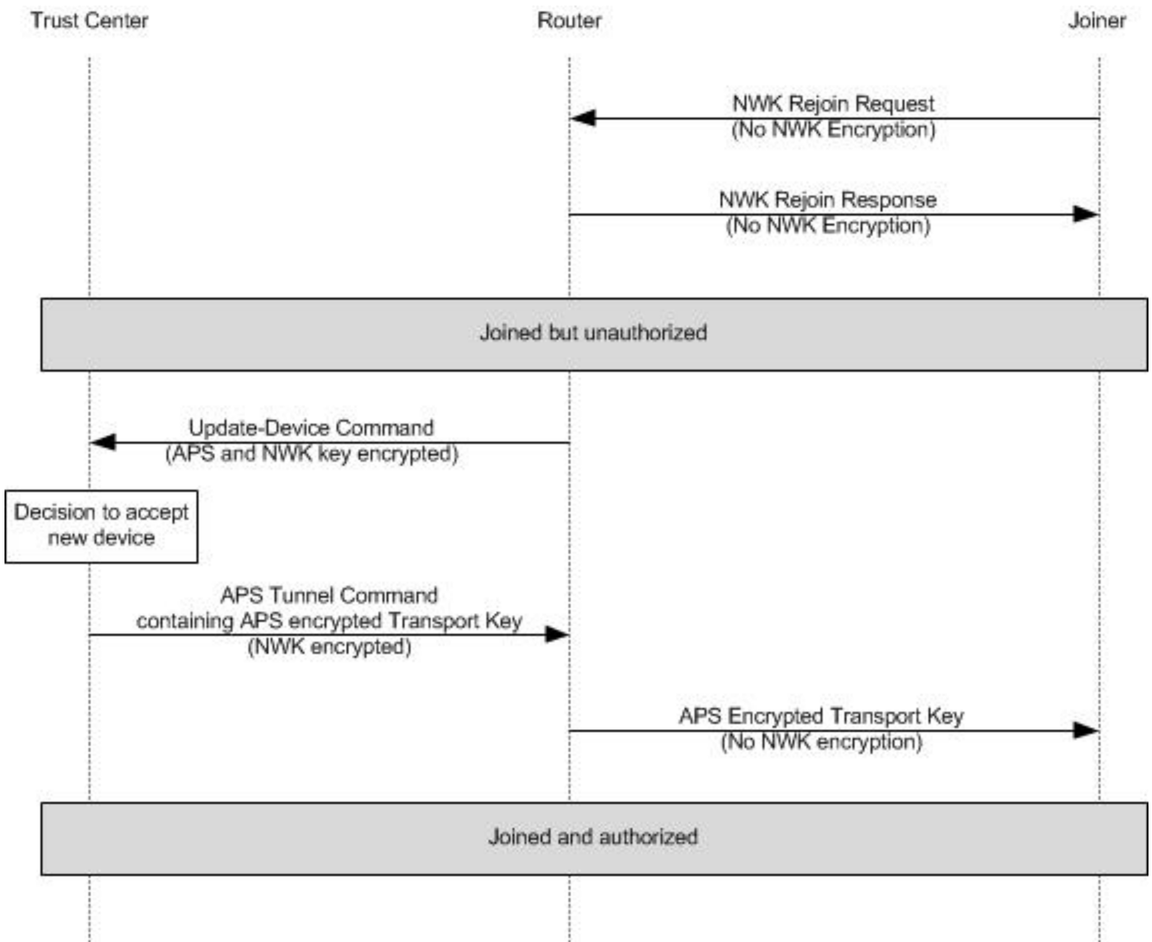


Figure 4-30. Trust Center Rejoin

A Trust Center Rejoin SHALL only be allowed in a centralized security network. Attempts to use a Trust Center rejoin in a distributed security network shall be rejected.

While it is conceptually possible to use a pre-configured link-key or security authentication token previously established during initial join to negotiate a new trust center link-key in the course of a trust center rejoin, and in particular the trust center swap-out procedure, this approach is not supported in current revisions of the specification. Therefore, a trust center SHALL NOT select a dynamic key negotiation scheme for a trust center rejoin. Trust center link-key updates SHALL be performed on-network, once the rejoin procedure completed successfully.

The trust center device SHALL assume no key agreement or pre-shared secret capabilities if the Supported Key Negotiation Methods Global TLV is not included in the network commissioning request. Prior knowledge of general key agreement capabilities, if any, SHALL be considered stale and ignored

The following sections describe the behavior of the devices in the network and the orphaned devices.

4.6.3.3.3 Coordinator and Router Operation

This text describes the security operations for support of rejoining which are to be carried out by the Zigbee coordinator and by Zigbee routers that are already operating on the network. These devices will receive rejoin requests by orphaned devices and will act as follows.

Following the steps described in section , an orphaned device (router or end device) SHALL be provisionally accepted onto the network by the coordinator or router for at least *apsSecurityTimeOutPeriod* milliseconds. During this period it SHALL be required to send at least one correctly formed Zigbee message secured with the network key to the new parent. If this message successfully passes all the security processing steps described in this document, it SHALL be accepted as a member of the network.

Starting from the time a device has been added to the nwkNeighborTable, after *apsSecurityTimeOutPeriod* milliseconds if that device does not send at least one network encrypted message, where it is using its long address in the network layer auxiliary security header, then it SHALL be deleted from the nwkNeighborTable. This applies regardless of whether the device type in the nwkNeighborTable is a router, coordinator, or end device

If the router, or Trust Center acting as router, receives an APSME-UPDATE-TUNNEL.indication then it SHALL reset the timeout for that specific device back to *apsSecurityTimeOutPeriod*. This allows the Trust Center to extend the timeout for the joining or rejoining device while it awaits user interaction or off-network verification to authenticate the device. This specification neither specifies nor requires any action from the router or coordinator in the case that a message from an orphaned device fails security processing above that required by text elsewhere in this document.

4.6.3.4 Network Key Update

The Trust Center and network devices SHALL follow the procedures described in this section when updating the active network key. Updating of the network key is not possible when operating in distributed security mode.

4.6.3.4.1 Trust Center Operation

When updating a standard network key with a new key of the same type, the Trust Center MAY broadcast or unicast the key update. If it chooses to broadcast the new key to all devices on the network it issues the APSME-TRANSPORT-KEY.request primitive with the DestAddress parameter set to the broadcast address and the StandardKeyType parameter set to 0x01 (that is, a network key).

For a unicast key update the Trust Center SHALL issue multiple APSME-TRANSPORT-KEY.request primitive with the DestAddress set to each device it wants to notify of the new key.

The TransportKeyData sub-parameters SHALL be set as follows for both unicast and broadcast key updates:

- The KeySeqNumber sub-parameter SHALL be set to the sequence count value for the new network key.
- The NetworkKey sub-parameter SHALL be set to the new network key.
- The UseParent sub-parameter SHALL be set to FALSE.

If the sequence count for the previously distributed network key is represented as *N*, then the sequence count for this new network key SHALL be (*N*+1) mod 256.

The Trust Center MAY cause a switch to this new key by issuing the APSME-SWITCH-KEY.request primitive with the DestAddress parameter set to the broadcast address and the KeySeqNumber parameter set to the sequence count value for the updated network key. The switch key SHALL NOT be unicast. It shall be encrypted at the network layer with either the current network key or the updated network key, and the key sequence number SHALL indicate the key used.

In centralized security mode, the Trust Center MAY maintain a list of all of the devices in the network. To update the active network key using this list, the Trust Center MAY first send the new network key to each device on this list and then ask the network to switch to the new key.

4.6.3.4.2 Network Device Operation

Devices SHALL be capable of storing 2 network keys, the current and an alternate.

When in the normal operating state and upon receipt of a APSME-TRANSPORT-KEY.indication primitive with the StandardKeyType parameter set to 0x01 (that is, a network key), a device SHALL accept the TransportKeyData parameters as a network key only if the SrcAddress parameter is the same as the Trust Center's address (as maintained in the *apsTrustCenterAddress* attribute of the AIB). If accepted, the key and sequence number data contained in the TransportKeyData parameter SHALL replace the alternate network key. Otherwise, the key and sequence number data contained in the TransportKeyData parameter SHALL replace the active network key. In either case, all incoming frame counters and the outgoing frame counter of the appropriate network key SHALL be set to 0.

When in the normal operating state and upon receipt of an APSME-SWITCH-KEY.indication primitive, a device SHALL switch its active network key to the one designated by the KeySeqNumber parameter only if the SrcAddress parameter is the same as the Trust Center's address (as maintained in the *apsTrustCenterAddress* attribute of the AIB). Figure 4-31 illustrates the procedure.

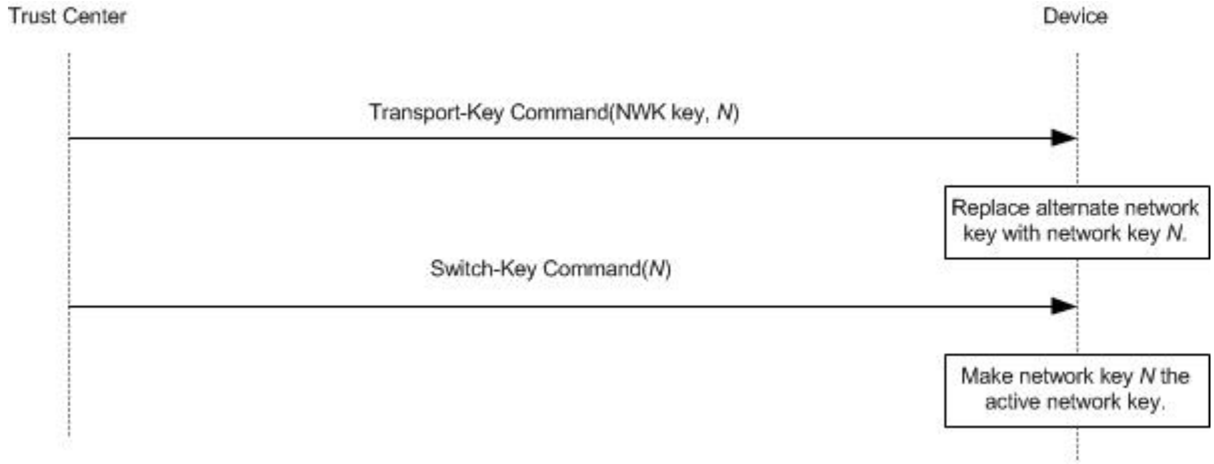


Figure 4-31. Example Network Key-Update Procedure

4.6.3.4.3 Message Sequence Chart

An example of a successful network key-update procedure for two devices is shown in Figure 4-31. In this example, the Trust Center sends a network key with sequence number *N* to the device. All devices are required to be capable of storing two network keys, an active and alternate. Upon receipt of the transport-key command, the device replaces its alternate network key with the new network key. Next, upon receipt of the switch-key command, the device makes the new network key the active network key.

4.6.3.5 Dynamic Link Key Negotiation

The Trust Center MAY provide support for dynamically negotiating a link key via the ZDO Security Key Negotiation services. If both trust center and partner device support key negotiation the joiner SHALL use one of the defined ways that key negotiation.

1) Prior to joining or rejoining, the Trust Center sends a ZDO Security Start Key Update Request to the partner device with the key negotiation method and, as far as applicable, the pre-shared secret it requests the partner device to use. The Trust Center will make use of the APSDE-DATA.request with Relay=TRUE to have the parent router relay the messages to the not yet joined device.
2) During normal operation on the network, the Trust Center sends a Security_Start_Key_Update_req to the partner device with the key negotiation method and, as far as applicable, the pre-shared secret it requests the partner device to use. When the partner device is a sleepy end device (RxOnWhenIdle=FALSE), the Trust Center SHOULD use an application defined mechanism to ensure that the partner is awake when it initiates the Dynamic Link Key Negotiation.
3) Immediately after joining or rejoining, the partner device discovers via the ZDO Node Descriptor Response that the Trust Center is capable of performing key negotiation and which key negotiation method and, as far as applicable, the pre-shared secret the partner device is EXPECTED to use.
    a) This could occur when the parent router does not support Revision 23 or higher of this specification and the device joined with the well-known link key or install code derived key.

Once the partner device has determined it should initiate key negotiation, it SHALL do the following.

1) Generate a ZDO Security Start Key Negotiation Request as described in section 2.4.3.4.6.
2) Wait up to *apsSecurityTimeOutPeriod* to receive a ZDO Security Start Key Negotiation Response. Processing SHALL be done as described in section 2.4.4.4.1.4.
3) If processing of the response was successful, generate an APSME-VERIFY-KEY.request.

4) Wait up to *apsSecurityTimeOutPeriod* to receive an APSME-CONFIRM-Key.indication that indicates the key was successfully confirmed.

The Trust Center SHALL do the following.

1) Process a request to start key negotiation as described in section 2.4.4.4.1.4. Successful processing will generate a ZDO Security Start Key Negotiation Response.
2) Wait up to *apsSecurityTimeOutPeriod* to receive an APSME-VERIFY-KEY.indication from the partner device.
3) Generate an APSME- CONFIRM-KEY.request.

The initial negotiation of a link key MAY be anonymous. After updating the link key using key negotiation and replacing the passphrase, anonymous key negotiation SHALL NOT be used for any further updates. All dynamic key negotiation SHALL be authenticated from then on.

4.6.3.5.1 Rejoining Device Operation

Following the steps described in section , an orphaned device (router or end device) SHALL be provisionally accepted onto the network for at least *apsSecurityTimeOutPeriod* milliseconds. During this period, it SHALL be required to send at least one Zigbee message, secured with the network key to the new parent.

If the device receives any message from its router parent it SHALL extend its timeout back to *apsSecurityTimeOutPeriod* to allow the Trust Center more time to authenticate it. If no messages are received after apsSceruityTimeOutPeriod it SHALL leave the network. As normal, the device SHALL NOT accept an unsecured network key (having no NWK security) from the Trust Center.

Note that a Zigbee device MAY also carry out an orphan scan as described in section . In this case it SHALL, at this time, also follow the steps described in this sub-section.

While it is conceptually possible to use a pre-configured link-key or security authentication token previously established during initial join to negotiate a new trust center link-key in the course of a trust center rejoin, and in particular the trust center swap-out procedure, this approach is not supported in current revisions of the specification. Therefore, a rejoining device SHALL NOT select a dynamic key negotiation scheme for a trust center rejoin. Trust center link-key updates SHALL be performed on-network, once the rejoin procedure completed successfully. A rejoining device SHALL reject attempts to negotiate a dynamic link key during rejoin.

A rejoining device SHALL NOT include the Supported Key Negotiation Methods Global TLV. This will allow a future version of the specification to revisit this limitation and enable rejoining devices to indicate support.

4.6.3.5.2 End-to-End Application Key Establishment

An initiator device, a Trust Center, and a responder device can follow the procedures described in this section when establishing a link key for purposes of end-to-end application security between initiator and responder devices. This process involves requesting a partner link key from the Trust Center with a third party device.

If both devices support Dynamic Link Key they MAY use Request Key to establish an Application Link Key first. Afterwards the initiator can use the APSME-KEY-NEGOTIATE.request to derive a new application link key using the previously requested key as the passphrase during the key negotiation.

4.6.3.5.3 Device Operation

The initiator device SHALL begin the procedure to establish a link key with a responder device by issuing the APSME-REQUEST-KEY.request primitive. The DestAddress parameter SHALL be set to the address of its Trust Center, the RequestKeyType parameter SHALL be set to 0x02 (that is, application link key), and the PartnerAddress parameter SHALL be set to the address of the responder device.

In a distributed security network where there is not a trust center to authorize the distribution of application link keys, an initiator device MAY issue an APSME-TRANSPORT-KEY.request to a responder device based on application policies on the device.

4.6.3.5.3.1 Upon Receipt of a Link Key

Upon receipt of an APSME-TRANSPORT-KEY.indication primitive with the StandardKeyType parameter set to 0x03 (that is, application link key), a device MAY accept the TransportKeyData parameters as a link key with the device indicated by the PartnerAddress parameter only if the SrcAddress parameter is the same as the *apsTrustCenterAddress* attribute of the AIB. If accepted, the *apsDeviceKeyPairSet* attribute in AIB table will be updated. A key-pair descriptor in the AIB SHALL be created (or updated if already present) for the device indicated by the PartnerAddress parameter, by setting the DeviceAddress element to the PartnerAddress parameter, the LinkKey element to the link key from the TransportKeyData parameter, the Features & Capabilities element to the Features field of the Link-Key Features and Capabilities TLV (if present), and the OutgoingFrameCounter and IncomingFrameCounter elements to 0 unless the value is the same as the previous link key.

In the case of a distributed security network, a device MAY accept an APSME-TRANSPORT-KEY.indication primitive with the StandardKeyType parameter set to 0x03 (that is, application link key) from a partner device since no trust center exists. The device and this partner can then establish an application link key based on the application level policies of the device.

4.6.3.5.4 Trust Center Operation

Upon receipt of APSME-REQUEST-KEY.indication primitives with the StandardKeyType parameter set to 0x02 (that is, application link key).

The Trust Center SHALL issue two APSME-TRANSPORT-KEY.request primitives with the StandardKeyType parameter SHALL be set to 0x03 (that is, application link key). The first primitive SHALL have the DestAddress parameter set to the address of the device requesting the key. The TransportKeyData sub-parameters SHALL be set as follows:

- The PartnerAddress sub-parameter SHALL be set to the PartnerAddress sub-parameter of the APSME-REQUEST-KEY.indication primitive's TransportKeyData parameter.
- The Initiator sub-parameter SHALL be set to TRUE.
- The Key sub-parameter SHALL be set to a new key K (link key).
- The TLVs sub-field SHALL include a TLV with the Link-Key Features and Capabilities local TLV included and set to the same value stored in the Trust Center's apsDeviceKeyPairSet entry for the device identified by the PartnerAddress sub-parameter of the APSME-REQUEST-KEY.indication primitive's TransportKeyData parameter.

The key SHALL have been generated in a random fashion. The second primitive SHALL have the DestAddress parameter set to the PartnerAddress sub-parameter of the APSME-REQUEST-KEY.indication primitive's TransportKeyData parameter. The TransportKeyData sub-parameters SHALL be set as follows:

- The PartnerAddress sub-parameter SHALL be set to the address of the device requesting the key.
- The Initiator sub-parameter SHALL be set to FALSE.
- The Key sub-parameter SHALL be set to *K*.
- The TLVs sub-field SHALL include a TLV with the Link-Key Features.

4.6.3.5.5 Message Sequence Chart

An example message sequence chart of the end-to-end application key establishment procedure is shown in Figure 4-32. The procedure begins with the transmission of the request-key command from the initiator to the Trust Center.

The Trust Center SHALL now send transport-key commands containing the application link to the initiator and responder devices. Upon completion (or time-out), the status of the protocol is reported to the ZDOs of the initiator and responder devices. If successful, the initiator and responder will now share a link key and secure communications will be possible.

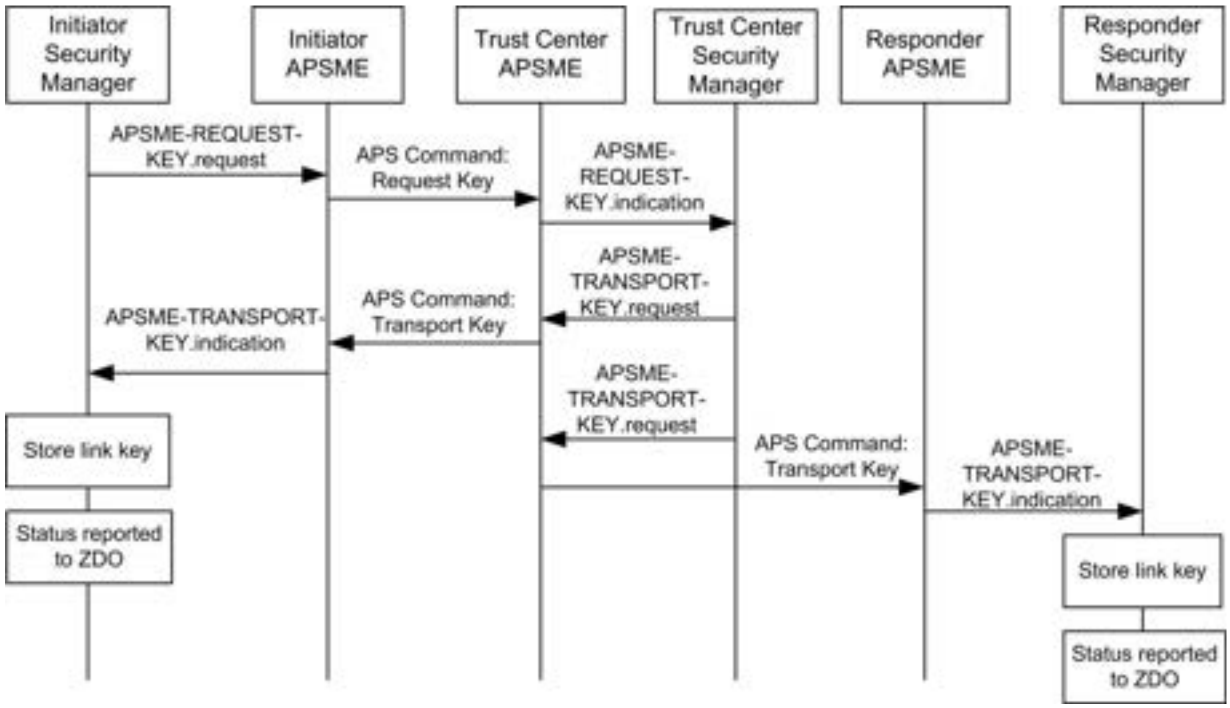


Figure 4-32. Example End-to-End Application Key Establishment Procedure

4.6.3.6 Network Leave

A device, its router, and the Trust Center SHALL follow the procedures described in this section when the device is to leave the network.

4.6.3.6.1 Trust Center Operation

If a Trust Center wants a device to leave and if the Trust Center is not the router for that device, the Trust Center SHALL issue the APSME-REMOVE-DEVICE.request primitive, with the ParentAddress parameter set to the router's address and the ChildAddress parameter set to the address of the device it wishes to leave the network.

The Trust Center will also be informed of devices that leave the network. Upon receipt of an APSME-UPDATE-DEVICE.indication primitive with the Status parameter set to 0x02 (that is, Device Left), the DeviceAddress parameter SHALL indicate the address of the device that left the network and the SrcAddress parameter SHALL indicate the address of parent of this device.

4.6.3.6.2 Router Operation

Routers are responsible for receiving remove-device commands and for sending update-device commands.

Upon receipt of an APSME-REMOVE-DEVICE.indication primitive, if the SrcAddress parameter is equal to the *apsTrustCenterAddress* attribute of the AIB then the command SHALL be accepted. The router SHALL ignore APSME-REMOVE-DEVICE.indication primitives with the SrcAddress parameter not equal to the *apsTrustCenterAddress* attribute of the AIB.

If the DeviceAddress corresponds to the local device's address, then the device SHALL remove itself from the network according to section 4.6.3.6.3. If the DeviceAddress corresponds to address of a child device then a router SHALL issue an NLME-LEAVE.request primitive with the DeviceAddress parameter the same as the DeviceAddress parameter of the APSME-REMOVE-DEVICE.indication primitive and the rejoin parameter set to 0. Other fields are defined by the stack profile.

If the DeviceAddress does not correspond to the local device address, nor does it correspond to a child device of the router, the command SHALL be discarded.

Upon receipt of an NLME-LEAVE.indication primitive with the DeviceAddress parameter set to one of its children and with the Rejoin Parameter = 0, a router that is not also the Trust Center SHALL issue an APSME-UPDATE-DEVICE.request primitive with:

- The DstAddress parameter set to the address of the Trust Center.
- The Status parameter set to 0x02 (that is, Device Left).
- The DeviceAddress parameter set to the DeviceAddress parameter of the NLME-LEAVE.indication primitive.

If the router is the Trust Center, it SHOULD simply operate as the Trust Center and SHALL NOT issue the APSME-UPDATE-DEVICE.request primitive (see section 4.6.3.6.1).

4.6.3.6.3 Leaving Device Operation

Devices are responsible for receiving and sending leave messages. The following rules apply to all three types of leave messages: NWK Leave Command, ZDO Mgmt Leave, and APS Command: Remove Device.

In a secured Zigbee network, leave messages SHALL be secured with the active network key and sent with security enabled at the level indicated by the *nwkSecurityLevel* attribute in the NIB.

In a secured Zigbee network, leave messages SHALL be received and processed only if secured with the active network key and received with security enabled at the level indicated by the *nwkSecurityLevel* attribute in the NIB.

A device SHALL only send a NWK leave message (request or announcement) if it has the active network key. A device that wishes to leave the network and does not have the active network key SHALL quietly leave the network without sending a NWK leave announcement.

4.6.3.6.4 Message Sequence Charts

Figure 4-33 shows an example message sequence chart in which a Trust Center asks a router to remove one of its children from the network. If a Trust Center wants a device to leave and if the Trust Center is not the router for that device, the Trust Center SHALL send the router a remove-device command with the address of the device it wishes to leave the network. In a secure network, the remove-device command SHALL be secured with a link key if present; otherwise SHALL be secured with the active network key. Upon receipt of the remove-device command, a router SHALL send a leave command to the device to leave the network.

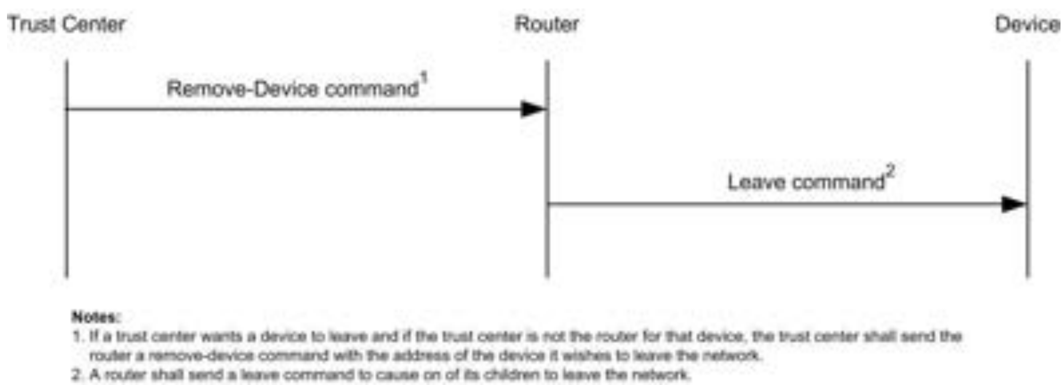


Figure 4-33. Example Remove-Device Procedure

Figure 4-34 shows an example message sequence chart whereby a device notifies its router that it is leaving the network. In this example, the device sends a leave command (secured with the active network key) to its router. The router then sends an update-device command to the Trust Center. In a secured network, the update-device command SHALL be secured with the link key, if present, or the active network key.

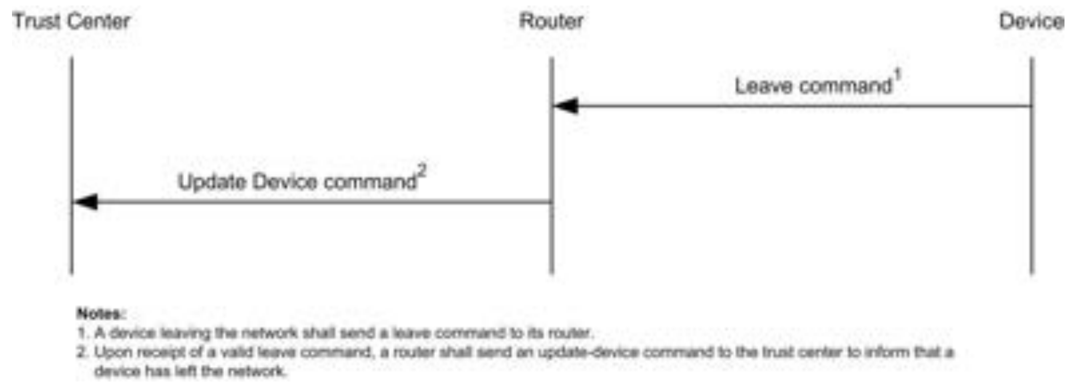


Figure 4-34. Example Device-Leave Procedure

4.6.3.7 Command Tunneling & APS Relaying

There are two commands used for over-the-air transportation of data to a node in the process of joining or rejoining. The original APS Command: Tunnel Data is used as a one-way communication of the network key to the device. This operation is called Command Tunneling.

To avoid problems of backward compatibility with the existing single-purpose tunnel command a new command has been created that is designed from the start to allow bi-directional communication: the APS Relay command. This operation is called Relaying.

Devices SHALL follow the procedures described in this section to allow secure communication between the Trust Center and a remote device that does not have the current network key.

4.6.3.7.1 Trust Center Operation

To embed a command in a tunnel command, the Trust Center SHALL first apply security protection as specified in section 4.4.1.1 and then, if security processing succeeds, the secured command frame SHALL be embedded in a Tunnel command frame as follows:

1. The APS header fields SHALL be set to the values of the APS header fields of the command to be embedded.
2. The destination address field SHALL be set to the 64-bit extended address of the destination device.
3. The tunneled auxiliary frame field SHALL be set to the auxiliary frame of the secured command, with following changes:
   - The extended nonce sub-field SHALL be set to 1.
   - The source address field SHALL be set to the 64-bit extended address of the Trust Center.
   - The tunneled command SHALL be set to the secured payload of the embedded command.
   - The tunneled command SHALL then be sent to the parent or other neighbor of the destination device.

4.6.3.7.2 Parent Operations

Upon receipt of an APS tunnel command, a router SHALL extract the embedded command as follows:

1. The APS header fields SHALL be set to the values of the APS header fields of the tunnel command.
2. The auxiliary frame field SHALL be set to the value of the tunneled auxiliary frame field of the tunnel command.
3. The APS payload field SHALL be set to the tunneled command field of the tunnel command.

The extracted command SHALL be sent to the destination indicated by the destination address field by issuing the NLDE-DATA.request primitive with security disabled.

4.6.3.7.3 Tunneled Data Destination Operation

The following applies to the end destination of the tunneled data payload after the parent has extracted and transmitted the payload from the APS tunnel command. Upon receipt of a message secured at the APS layer and with an extended nonce in the APS auxiliary frame, the message SHALL be processed as usual, except that the message SHALL NOT be looked up in, or added to, the APS duplicate rejection table.

4.6.3.7.4 Multi-hop with Dynamic Key Negotiation Joining

In order to facilitate a node joining and negotiating a link key multiple hops away from the Trust Center it is necessary to exchange a number of messages between the joiner and Trust Center. Since the joiner has not yet been authorized, the network SHALL rely on the parent router to relay the messages from the joiner in a way that restricts communication to only the Trust Center. If the Trust Center conditionally accepts the new device then the router will accept new messages and relay them to the trust center with limited filtering of those packets. The limited filtering is meant to future proof communication between trust center and joiner so that the router does need to be upgraded to support newer devices that MAY send different messages than the router knows about.

4.6.3.7.5 Tunneling and Relaying of Messages

The APS Tunnel Command is limited to a single use case of sending from the Trust Center through the parent router to the Joiner. It does not handle tunneling packets upstream to the Trust Center. To avoid problems of backward compatibility with the existing single-purpose tunnel command a new command has been created that is designed from the start to allow bi-directional communication: the APS Relay command.

Before key negotiation messages are relayed through the parent router, the joiner's own capabilities SHALL be relayed to the Trust Center and the Trust Center acknowledges back to the joiner what the next steps are. The Network Commissioning Request Command Frame will contain the joining device's capabilities as TLVs inside the Joining Device Encapsulation Global TLV. This specific TLV will be picked up by the parent router and relayed in the APS Update Device Command Frame.

A legacy Trust Center will ignore the TLV data and perform its normal legacy processing of the request. If the device is allowed to join the Trust Center will send the current network key in an APS Transport Key Command encrypted with the device's link key, and tunnel that message inside an APS Tunnel Command.

A newer Trust Center will examine the capabilities of the joining device as relayed in the Joining Device Encapsulation Global TLV. If the device is being allowed to join but is an older device, or one without Key Negotiation capabilities, the Trust Center will behave much like a Legacy Trust Center. If the device and Trust Center have Key Negotiation Capabilities the Trust Center will send a ZDO Security_Key_Update_req command to signal that the device is allowed to go ahead and start key negotiation. The ZDO command will be contained inside the APS Relay Command.

The Trust Center and Joining Device will continue to exchange messages back and forth through the parent router. The messages are all APS Command Relay Downstream and APS Relay Command Upstream containing the actual message that will be forwarded to the target. These messages MAY be fragmented and MAY enable APS Retries. The APS Acknowledgements will also be forwarded between Trust Center and Joiner via APS Relay Command frames. The APS Command Relay Downstream and APS Relay Command Upstream themselves will not be fragmented, and retries MAY be enabled on those messages. This avoids multiple levels of fragmentation on the same underlying message.

Fragmentation parameters are communicated using the Fragmentation Parameters Global TLV. The joiner includes this TLV inside the Joining Device Encapsulation Global TLV. That TLV is sent to the parent router via the Network Commissioning Request Command Frame, and relayed to the Trust Center in the APS Update Device command. Fragmentation Parameters of the Trust Center are communicated to the joiner via the ZDO Security_Key_Update_req that is contained within an APS Relay Command.

Key Negotiation is done via the ZDO Security_Start_Key_Negotiation_req and ZDO Security_Start_Key_Negotiation_rsp. The new key is verified by both sides using the APS Verify Key and APS Confirm Key Messages. Once the key is verified both sides can use the key but the device is not yet on the network.

The Trust Center MAY abort the communication at any time by issuing an APS Remove Device to the parent router. At that point the parent router will no longer accept messages from the Joining Device that need to be relayed.

After a key is negotiated but before the Trust Center grants access the Trust Center MAY exchange additional messages with the device as part of the Device Interview. This step is optional and the choice to use it is determined by the Trust Center. Those messages are also sent via APS Relay Command frames and all Device interview messages are APS encrypted with the newly negotiated link key.

Once the Trust Center decides to grant access it uses the same mechanism as a Legacy Trust Center. It will send the current network key in an APS Transport Key Command encrypted with the device's link key, and tunnel that message inside an APS Tunnel Command. In this case the link key used will be the one that was negotiated via ZDO.

4.6.3.7.6 Parent Router Filtering

By default, a Parent router will not relay frames from an unauthorized joining device. A router keeps track of unauthorized joiners in the nwkNeighborTable by setting the Relationship status of that device to 0x05 unauthenticated child.

The router waits to receive an APS Relay Command from the Trust Center addressed to that joining device. At that point it sets the status of the child to 0x06, unauthenticated child with relay allowed.

The router will keep track of each unauthorized node independently in case multiple devices join through the parent router.

A router will only accept APS Command Relay Downstream and APS Relay Command Upstream from the joiner if its status is 0x06, unauthenticated child with relay allowed. The router parent will maintain a security timer for each of those devices. If they do not become fully authorized on the network before the timer runs out, the device is removed from the neighbor table. The timer is reset only when the Trust Center sends an APS Relay command; it is never reset by the joining device.

4.6.3.7.7 Tunneling and Relaying of APS Data and APS Commands

The APS Tunnel command SHALL always be used to tunnel the APS Command: Transport Key from the Trust Center to a joining or rejoining device. The APS Tunnel Command SHALL NOT be used for tunneling any other APS data or APS command. The APS Tunnel Command SHALL be used for the APS Transport key even when both the Trust Center and joiner/rejoining device support Revision 23 and the APS relay command.

The APS Relay command SHALL be used for sending and receiving all other messages to a joining or rejoining device while it is not on the network. If an parent router is needed to send over multiple hops it also needs to be R23 in order to relay messages.

The APS relay command SHALL be used to encapsulate messages from the relaying router. This SHALL occur even in the circumstance that the relaying router is also the Trust Center. In other words, the APS relay command SHALL be used for single-hop or multi-hop join/rejoin situations.

An example of a full multi-hop join using the APS Relay command can be seen in Figure 4-35.

Figure 4-35. R23 Joining Using Multi-hop Relay

4.6.3.7.8 APS Retries for Relay Commands

APS retries MAY be used on the APS Command Relay Downstream and APS Relay Command Upstream themselves. APS retries can also be used on the messages within the APS Relay Commands.

4.6.3.7.9 Encryption of Relay Commands and their contents

APS encryption SHALL NOT be used on the APS Relay command itself. The APS message within the APS Relay will use APS encryption as required by this specification or the Application.

Figure 4-36 is an example of APS encryption of the APS Command Confirm Key sent from the Trust Center to the intermediate Router for relay to the joining / rejoining device.

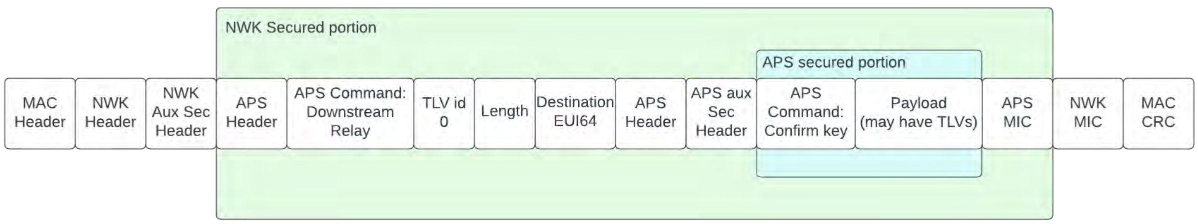


Figure 4-36. Relay Command Frame to the Parent Router

When the message is sent from the intermediate router to the joining / rejoining device it will look very familiar but will NOT have Network Encryption. Figure 4-37 shows the message in that case.

Figure 4-37. Relay Command Frame to the Joiner

4.6.3.7.10 Message Size and Fragmentation for APS Relayed Messages

When relaying messages prior to joining or rejoining, the Trust Center and Joiner can use fragmentation and APS Acknowledgements. Each side SHALL advertise fragmentation parameters via the Fragmentation Parameters Global TLV.

The APS Command Relay Downstream and APS Relay Command Upstream SHALL NOT be fragmented. The messages inside the APS Relay Command MAY be fragmented when needed.

When encapsulating a message inside an APS Relay command the joiner SHALL take into consideration the following when calculating the maximum size of the message within the APS relay command.

1. Overhead of the APS Relay command itself
2. Overhead of NWK Auxiliary Security Header and NWK Auxiliary Security Header MIC

The joiner SHALL not include the NWK Auxiliary Security Header and NWK Auxiliary Security Header MIC when sending to the parent router. However, the joiner SHALL leave space in the message for the parent router to add these headers. The Maximum size of an APS Relayed message is shown in Table 4-40.

Table 4-40. Maximum Message Size of an APS Relayed Message

Item	Size (bytes)	Notes
Max 802.15.4 MTU	128	
PHY Length byte	1	
MAC Header	9	
NWK Header	8	
NWK Auxiliary Security Header	14	The size of this overhead is always included even when the message is not NWK encrypted.
APS Header for APS Command	2	
APS Relay Overhead	10	
NWK Auxiliary Security Header MIC	4	The size of this overhead is always included even when the message is not NWK encrypted.
MAC CRC	2	
Max Remaining size of APS Relayed Message	79	The APS relayed message SHALL include its own APS Header. **Note:** If source routing is enabled on the TC, the maximum remaining size of the APS relayed message would be smaller. The network header will have the source route present and depending on the number of hops the payload will be reduced.

If the parent router receives an APS Relay command from a joiner that would exceed the max MTU when NWK Encryption is applied, the parent router SHALL drop the message and no further processing SHALL be done.

When encapsulating a message inside an APS Relay command the Trust Center SHALL take into consideration the following overhead when calculating the maximum size of the message within the APS relay command.

1. Overhead of the APS Relay command itself
2. Overhead of NWK Auxiliary Security Header and NWK Auxiliary Security Header MIC
3. The size of any source routing overhead in the NWK Header if source routing is used.

If an APS Datagram to be relayed is greater than the max MTU then it SHALL be fragmented, and the fragments SHALL be put inside APS Relay messages. Both Trust Center and Joiner are required to support fragmentation starting with R23.

If an APS Command to be relayed is greater than the max MTU then it SHALL be dropped. No APS command for this specification will exceed the max MTU when relayed.

When sending APS Acknowledgements for relayed message fragments, the APS Acknowledgements SHALL also be sent within APS Relay Commands.

Figure 4-38 and Figure 4-39 show how fragments are relayed inside the APS Command. This is the last hop between router parent and joiner/rejoiner, with no network encryption.

Figure 4-38. Relay Frame with Fragmentation to the Parent Router

Figure 4-39. Relay Frame with Fragmentation to the Joiner

4.6.3.7.11 Rules for Processing Relay Commands for already joined devices

This section applies to devices that are NOT currently joining or rejoining the network. This section details the rules for forwarding and processing relay messages.

Upon receipt of an APS_CMD_RELAY_MESSAGE_DOWNSTREAM the router SHALL do the following.

1. If the message was not NWK encrypted it SHALL be dropped and no further processing SHALL be done.
2. If the message is not from the Trust Center it SHALL be dropped and no further processing SHALL be done.
3. Execute the General Processing Rules in Annex I. If the result indicates an error, then the message SHALL be discarded and no more processing SHALL be done.
4. If the message does not contain the Relay Message TLV then it SHALL be discarded, and no more processing SHALL be done.
5. Examine the Destination EUI64 in the Relay Message TLV.
   a. If there is no entry in the *nwkNeighborTable* with an Extended Address matching the Destination EUI64 then the message SHALL be dropped, and no further processing SHALL be done.
   b. If a matching entry is found and the Relationship is 0x05, unauthorized child, then the Trust Center has given permission for relay to occur. The device SHALL do the following:
      i. Perform an NLME-SET.request on the *nwkNeighborTable* for that entry. Set the Relationship status to 0x06, unauthorized child with relay allowed.
   c. If a matching entry is found and the Relationship is neither 0x05 (unauthorized child) nor 0x06 (unauthorized child with relay allowed), discard the message and no further processing SHALL be done.
   d. Update the security timer for how long the device is allowed to be an unauthorized neighbor. The router SHALL do the following:
      i. Perform an NLME-SET.request on the *nwkNeighborTable* for that entry, set the SecurityTimer field to *apsSecurityTimeOutPeriod*.
   e. Transmit the data in the message Message to be Relayed field of the Relay Message TLV to the device associated with the nwkNeighborTableEntry. Create a new APS Command Header with Command ID APS_CMD_RELAY_MESSAGE_DOWNSTREAM with a new APS sequence number generated by the parent router. Append the complete Relay Message TLV from the received APS_CMD_RELAY_MESSAGE_DOWNSTREAM command frame. Generate an NLME-DATA.request with the new APS Command header and the APS_CMD_RELAY_MESSAGE_DOWNSTREAM message.

Upon receipt of an APS_CMD_RELAY_MESSAGE_UPSTREAM the router or coordinator router SHALL do the following:

1. If the device is a router and the message was NWK encrypted it SHALL be dropped and no further processing SHALL be done. Routers SHALL ignore these messages from other devices on the network.
2. If the outer APS command was APS encrypted it SHALL be dropped and no further processing SHALL be done.
3. Execute the General TLV Processing Rules in Annex I. If the result indicates an error, then the message SHALL be discarded and no more processing SHALL be done.
4. If the message does not contain the Relay Message TLV then it SHALL be discarded, and no more processing SHALL be done.
5. Examine the Source EUI64 in the Relay Message TLV.
    a. If there is no entry in the *nwkNeighborTable* with an Extended Address matching the Source EUI64 then the message SHALL be discarded and no further processing SHALL be done.
    b. If a matching entry is found and the Relationship is not 0x06, unauthorized child with relay allowed, then the message SHALL be dropped, and no further processing SHALL be done.
    c. Send the entire received Relay Message TLV to the Trust Center. Create a new APS Command Header with Command ID APS_CMD_RELAY_MESSAGE_UPSTREAM, with a new APS sequence number generated by the parent router. Append the complete Relay Message TLV from the received APS_CMD_RELAY_MESSAGE_UPSTREAM command frame. The device SHALL then execute an NLDE-DATA.request with the following parameters. If the device is the Coordinator this will result in a loopback message.
        i. The NDSU SHALL be the new APS Relay Upstream Command including APS Header.
        ii. DstAddrMode SHALL be set to 0x02, 16-bit network address.
        iii. DstAddr SHALL be set to 0x0000.
        iv. SecurityEnable SHALL be set to TRUE.
        v. UseAlias SHALL be set to FALSE.

It is important to note that unknown TLVs in the APS Command Relay Upstream are NOT relayed by the parent router. Only known TLVs described in this section are relayed to the TC or to the Joiner. The only known TLV in this specification revision is the Relay Message TLV. All unknown TLVs inside the APS Command Relay Downstream and APS Relay Command Upstream are dropped by the parent router. If Trust Center and Joiner wish to include newer TLVs unknown to the parent router they must be put inside the message that is embedded within the Relay Message TLV. TLVs inside the Relay Message TLV SHALL not be parsed by the parent router and passed thru in their entirety.

4.6.3.7.12 Rules for Processing Relay Commands for Joining or Rejoining Devices

Devices joining or rejoining the network need to protect against unencrypted messages they receive during their attempt to get onto, or back onto, the network. Filtering the messages is key part of this protection.

Devices joining or rejoining the network SHALL only accept APS Relay Downstream Commands, and APS Transport Key commands from their router parent that are *unencrypted* at the network layer.

Before a link key has been successfully negotiated the stack SHALL not pass up messages to the application layer.

4.6.3.7.13 Device Interview Timeouts and Exiting Device Interviews

After a key has been negotiated, but before the device has received APS Transport Key command, there is a period of message exchange known as the Device Interview. The Trust Center decides whether this step is performed. During the Device Interview the stack SHALL reject messages inside the Relay Message TLV that do not have APS encryption with the newly negotiated key.

The stack SHALL pass up APS encrypted application layer messages to the application layer for processing during the device interview.

During the Device Interview the application layer MAY restrict the application layer messages that are allowed to be sent or received. It could do this, for example, by restricting the cluster ID of messages inside the Relay Message TLV of the APS Relay Downstream and APS Relay Upstream commands. Again, these messages are still required to be APS encrypted.

Inter-PAN messages are not permitted during device interview process. If received, they SHALL be dropped. Messages that do not have Delivery Mode of Normal Unicast Delivery SHALL be dropped. The ZDO messages are normally NWK encrypted only. During device interview , these messages SHALL be APS encrypted. If these are NWK encrypted, they SHALL be dropped.

The following client requests are allowed during device interview, the destination addressing SHALL be unicast only.

- NWK_addr_req
- IEEE_addr_req
- Node_Desc_req
- Power_Desc_req
- Simple_Desc_req
- Active_EP_req
- Match_Desc_req

All other ZDO requests shall be dropped and it is permissible to send a NOT SUPPORTED response for everything that is not allowed during device interview.

The Device Interview period is complete once the Trust Center chooses to allow the device on the network and sends the APS Transport Key command, or the Trust Center chooses to reject the joining/rejoining device. Once the Device Interview period is complete and the device is on the network the application's normal rules for processing messages will apply.

Before establishing a TCLK the Joining Device and parent router use the apsSecurityTimeOut to determine how long to wait for a message before giving up and removing all entries in the neighbor table. After Establishing the TCLK the joiner and the trust center operate on a "rolling timeout" that periodically refreshes on device interview stack events.

Trust center refreshes the timeout for a joining device by apsDeviceInterviewTimeoutPeriod whenever it generates a downstream request.

4.6.3.7.14 Device Interview Timeouts and Exiting Device Interviews

Parent router device refreshes the timeout for a joining child by apsDeviceInterviewTimeoutPeriod whenever there is a downstream relay command with network encryption for one of its joining children.

Joiner refreshes its timeout by apsDeviceInterviewTimeoutPeriod when it receives a downstream relay command with a relay payload encrypted with the negortiated TCLK.

If a device interview session is quiet for apsDeviceInterviewTimeoutPeriod and the TC has not sent the APSME Transport Key command with the Network Key to the joining device the joining device will fail the joining process. If the session apsDeviceInterviewTimeoutPeriod for the joining device expires on the TC and Parent router, without receiving network encrypted traffic from the joiner, the Device Interview session will be closed. The TC will clear the associated apsDeviceKeyPairTable entry and notify the higher layer with a SECURITY_FAIL (0xAD) error. the joining device will need to issue a separate nwk commissioning join request to establish a new TCLK.

4.6.3.8 APS Frame Counter Verification

All devices implementing Revision 23 of this specification SHALL support enforcement of verified APS frame counters. A verified frame counter is where the receiver of an APS encrypted frame has knowledge of a recent frame counter that was successfully received.

Under certain circumstances a device can lose track of the latest APS frame counter of the corresponding device in the apsDeviceKeyPair entry. When this occurs, the device SHALL set VerifiedFrameCounter to FALSE for the corresponding entry of the apsDeviceKeyPairSet.

One example of losing track of the latest frame counter is after a reboot. After reboot a device commonly sets the IncomingFrameCounter for all apsDeviceKeyPairSet entries to 0.

Since a device does not know how long it has been down after a power cycle, it is required to verify the frame counter for other devices it shares a link key with. After reboot, all entries in apsDeviceKeyPairSet SHALL set VerifiedFrameCounter to FALSE.

If the result of the Security Processing of Incoming Frames is UNVERIFIED_FRAME_COUNTER then the received message is dropped. Frame Counter synchronization SHALL be initiated to the sender of the message by following the procedure in section 4.6.3.8.1, subject to rate limiting as described in section 4.6.3.8.1.

Figure 4-40 illustrates the general flow of how this will work.

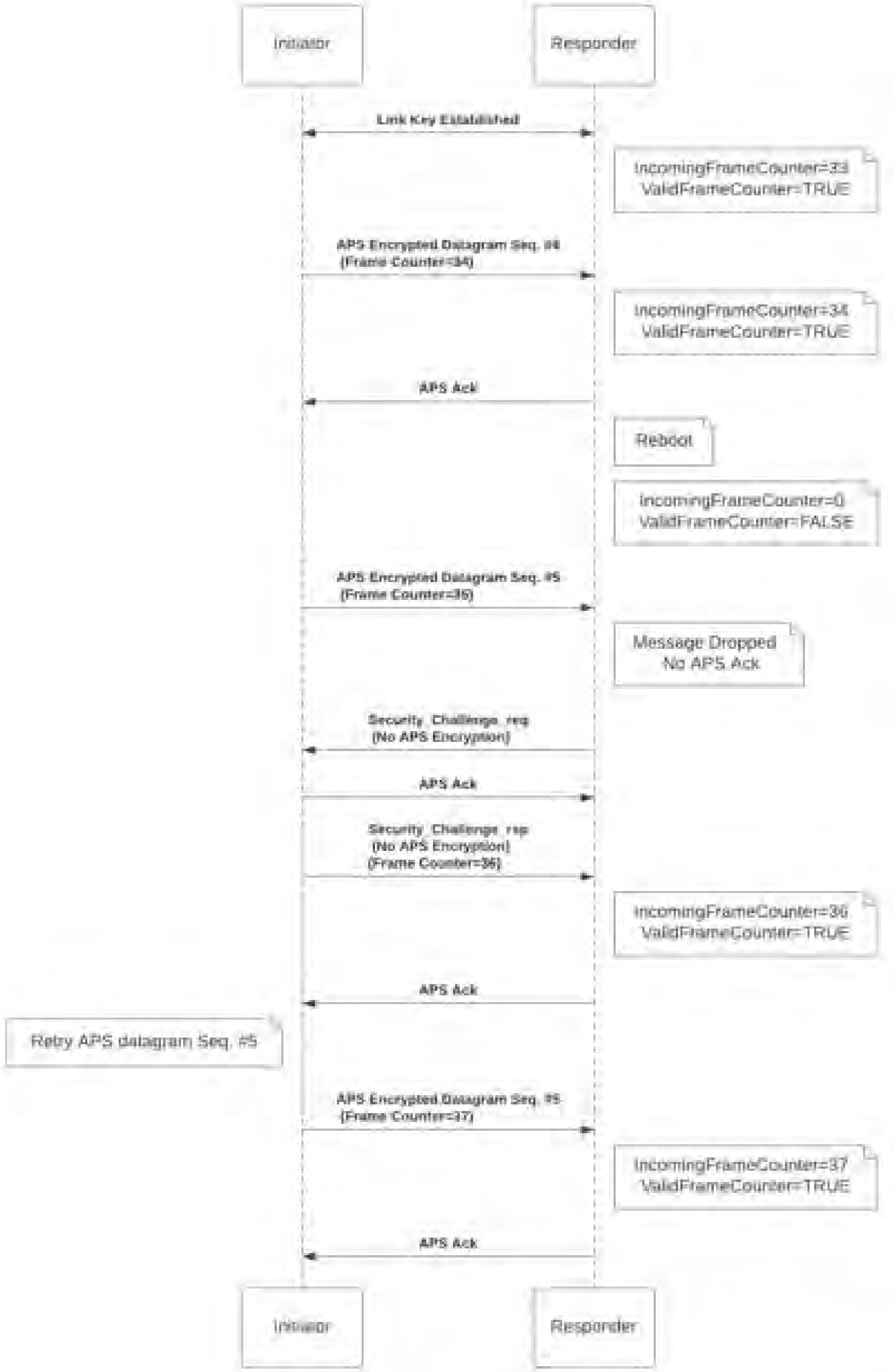


Figure 4-40. APS Frame Counter Synchronization

Table 4-41 indicates when to set the Verified Frame Counter Sub-field. Sending devices SHALL use this when determining what value to set.

Table 4-41. Setting for Verified Frame Counter Field based on APS Frame Type

Description	APS Frame Type Field (b1 b0)	ACK Format Field
APS Datagram	00 (Data)	1
APS Command	01 (Command)	0
ACK to Datagram	10 (ACK)	1
ACK to Command	10 (ACK)	0

When the apsDeviceKeyPairSet entry indicates VerifiedFrameCounter is FALSE and the Frame Counter Synchronization bit in the Features & Capabilities bitmap = '1', frame counters from received APS encrypted messages SHALL not be stored. Legacy devices will not have Frame Counter Synchronization bit in the Features & Capabilities bitmap = '1', and as a result the incoming frame counter for APS encrypted messages MAY be stored. The rules for processing incoming messages are detailed in section 4.4.1.2.

It is permissible for a device to proactively synchronize unverified frame counters prior to receiving an APS encrypted message that would trigger synchronization. This could be done by periodically examining its *apsDeviceKeyPairSet* for entries where VerifiedFrameCounter is FALSE and following the procedure in section 4.6.3.8.1.

After a device initially joins a network by receiving an APS Encrypted Transport Key command, it SHALL set the VerifiedFrameCounter to TRUE and the IncomingFrameCounter field to the frame counter value for the received APS Transport Key Command. Synchronization is not necessary because there are no previous messages that can be replayed to the device.

If a device rejoins a network, it SHALL NOT change the state of VerifiedFrameCounter for its *apsDeviceKeyPairSet* entries. A rejoining device initiates frame counter synchronization based on subsequent APS encrypted messages it receives and the state of its *apsDeviceKeyPairSet*, as described previously in this section.

4.6.3.8.1 Initiating a Challenge

When a device determines it needs to send an APS Frame counter challenge it SHALL do as follows.

1) Generate an 8-byte random challenge value.
2) Construct an APS Frame Counter TLV with the Sender's EUI64 and the random value.
3) Store the challenge in *apsChallengeValue* of the AIB
4) Store the target EUI64 of the challenge in the apsChallengeTargetEui64.
    a) This SHALL be the same value as the DeviceAddress of the associated *apsDeviceKeyPairSet* entry.
5) Set the *apsChallengePeriodRemainingSeconds_ of the AIB to _apsChallengePeriodTimeoutSeconds.*
6) Start a timer that decrements apsChallengePeriodRemainingSeconds_ every second.
7) Unicast the ZDO Security_Challenge_req to the *apsChallengeTargetEui64*.
    a) The message SHALL NOT apply APS encryption.

Implementations MAY provide contextual storage for apsChallengeTargetEUI64 to allow multiple challenge requests to different targets simultaneously.

4.6.3.8.2 Challenge Nonce and Key

To generate the challenge response the responder SHALL construct a nonce value as follows:

a) The Source EUI-64 set to the EUI-64 of the device originating the challenge response frame.

b) The frame counter field set to apsChallengeFrameCounter.

c) The security control frame set to reflect security level 2: No encryption, 64-bit MIC.

The responder SHALL then increment apsChallengeFrameCounter by one. apsChallengeFrameCounter is reset to 0 whenever the APS outgoing frame counter for that key is incremented, and upon reboot.

The AES-CCM-128 key for the challenge response is set to the link-key, where bytes 0, 4, 8, and 12 are XORed with the bytes 0, 1, 2, and 3 of the 32-bit APS outgoing security frame counter assuming a little-endian in-memory representation of that value. For example, if the APS outgoing security frame counter was 0x11223344, and stored in memory as { 0x44, 0x33, 0x22, 0x11 }, and the APS key was C0:C1:C2:C3:C4:C5:C6:C7:C8:C9:CA:CB:CD:CE:CF, the resulting AES-CCM-128 key for the challenge response would be calculated as: 84:C1:C2:C3:F7:C4:C5:C6:C7:EA:C9:CA:CB:DC:CE:CF. This allows multiple challenges for the same outgoing security frame counter to be answered without reusing the nonce for the same key.

Initiator and responder use the same approach to derive the symmetric AES-128-CCM key for the challenge/response exchange.

4.6.3.8.3 Computing the AES-CCM-128 MIC for a Challenge

To compute the AES-CCM-128 MIC for a particular combination of link-key, corresponding outgoing security frame counter, and challenge, the initiator and responder execute the CCM* authentication scheme as specified in Annex A applying security level 2: No Encryption, 64-bit Message Integrity Code (MIC). The CCM* procedure is executed with the following parameters:

1) a = the octet string that represents the Frame Counter Challenge TLV, including tag-ID and length fields, and the current APS frame counter, in little-endian representation
2) m = { } (empty octet string)
3) N = nonce as determined in section 4.6.3.8.2.
4) Key as determined in section 4.6.3.8.2.

4.6.3.8.4 Responding to a Challenge

1) Construct an APS Frame Counter Response TLV with the following:
    a) The local device EUI64 value as the Sender EUI64 field.
    b) The random value in the received APS Frame Counter Challenge TLV.
    c) The current value of the OutgoingFrameCounter for the corresponding *apsDeviceKeyPairSet* entry.
    d) The value of the apsChallengeFrameCounter that was used to calculate the AES-CCM-128 MIC for the response.
    e) The AES-CCM-128 MIC that was calculated for the particular combination for challenge, link-key and outgoing security frame counter.
2) Unicast the ZDO Security_Challenge_rsp to the Sender EUI64 value of the APS Frame Counter Challenge TLV.
    a) Set the Status to SUCCESS.
    b) Do not apply APS encryption.

4.6.3.8.5 Validating the Response to a Challenge

On receipt of a message containing the APS Frame Counter Response TLV, do the following.

1) Validate the Responder MIC field:
   a) Create the Responder Challenge Key value as described in section 4.6.3.8.2.
   b) Execute AES-CCM algorithm described in section .
   c) Compare the calculated MIC to the value of the Responder MIC field.
      i) If they do not match, no further processing SHALL be done.
         (1) If they do match set the corresponding apsDeviceKeyPairSet_ entry as follows:
            a) Set the IncomingFrameCounter to the APS Frame Counter value of the TLV.
            b) Set the VerifiedFrameCounter to TRUE.

4.7 Security Operations in Centralized Security Networks

The security services provided here offer a range of options within a Zigbee network. For interoperable and consistent field behavior, a more defined set of policies and processes is defined here. The basis for these operations is that the device forming a network can establish security policies believed appropriate for the network and that a joining device will acknowledge and use the policies in place in the network. Joining is therefore based on the forming device setting policies within the allowed settings in this section and the joining device having the appropriate flexibility to adapt to these settings.

4.7.1 Trust Center Policies

The Trust Center is a critical security component in a Zigbee network. The policies that the Trust Center puts in place control what devices get on the network and how they do so in a secure manner. Security is not an end unto itself but a means to establish a reasonable level of protection of the application and data that is being transmitted across the Zigbee network. Often an increase in security increases the overhead in management, requires additional time and functional states while security is negotiated, and can detract from a user experience by requiring them to go through additional steps that seem unnecessary. Therefore a balance SHALL be struck between the hardening the network against attacks and the ability to use the network for the applications it was intended for.

It is important to understand the security decisions that are being made in the network and the design of the Trust Center application is at the heart of those decisions. This section presents the options and settings for the Trust Center and requires a series of choices to be set on network start up.

4.7.2 Trust Center Link Keys

Support for link keys SHALL be required for all devices. Link keys offer an additional level of security for devices to be able to send messages with end-to-end security instead of just with the hop-by-hop security provided by network encryption.

In addition, link keys are crucial for providing a simple authorization mechanism. The Trust Center can send devices a copy of the network key that is intended only for a specific device using that device's link key to encrypt the message.

4.7.2.1 Trust Center Passphrase Updates

When a device is initially added to the apsDeviceKeyPairEntrySet the PassphraseUpdateAllowed is set to TRUE. After successfully joining and negotiating a link key the device is required to update its authentication token (symmetric passphrase). After update, the PassphraseUpdateAllowed field for that device's entry is set to FALSE and is never set back to TRUE. The only way to change the passphrase at that point is to administratively remove the apsDeviceKeyPairEntrySet on the Trust Center and perform a fresh join to the network for that device.

Future revisions of this specification may expand the set of authentication token types and will need to describe how those tokens are deployed and the interaction with the existing token type.

4.7.3 Trust Center Policy Values

The following is a list of configuration values that relate to the Trust Center's policy decisions that are part of the security related AIB in Table 4-35. They will be used to describe requirements for dictating the network security policies. The trust center can use these policies to create higher or lower sets of security and controls on the network. For example:

- A system can be set up with centralized security such that any device can join the network. In such a permissive network, trust center link keys are still updated from the global value used initially for joining.
- A system can also be set up with trust center policies that only allow known devices. A user SHALL then install the IEEE address and a link key for the new device into the trust center prior to the device joining. This could be done using install code based keys. This validates to the joining device that it is on a network that knows its identity during the joining process. The trust center in this network can also update the trust center link keys of joining devices so secure key updates and rejoining can be conducted during the lifetime of the network.

Table 4-42 describes the Trust Center policy values trustCenterPolicies of the AIB and their usage.

Table 4-42. Trust Center Policy Values

Attribute	ID	Type	Range	Description	Usage
allowJoins	0xad	Boolean	TRUE or FALSE	This boolean indicates whether the Trust Center is currently allowing devices to join the network. A value of TRUE means that the Trust Center is allowing devices that have never been sent the network key or a trust center link key, to join the network.	This is set to FALSE in centralized security networks that do not want to allow new devices on the network.
requireInstallCodesOrPresetPassphrase	0xaf	enumeration	0x00 – 0x10	This enumeration indicates if the Trust Center requires install codes to be used with joining devices. 0x00 – do not support Install Codes 0x01 – Support but do not require use of Install Codes or preset passphrases 0x02 – Require the use of Install Codes by joining devices or preset Passphrases	This is set to 0x02 if the trust center requires install codes in new devices performing symmetric key joins or preset passphrases for key negotiations. Trust Centers that support setting 0x01 or 0x02 SHALL provide a user interface or out of band means to input the Install Code.
allowRejoinsWithWellKnownKey	0xb6	Boolean	TRUE or FALSE	This value indicates if the trust center allows rejoins using well known or default keys. A setting of FALSE means rejoins are only allowed with trust center link keys where the KeyAttributes of	By default, this attribute shall be set to FALSE. A higher-level Trust Center

Attribute	ID	Type	Range	Description	Usage
				the apsDeviceKeyPairSet entry indicates VERIFIED_KEY.	policy may change the value.[10]
allow-TrustCenterLinkKeyRequests	0xb7	Enumeration	0x00 – 0x02	This value controls whether devices are allowed to request a Trust Center Link Key after they have joined the network. It MAY have the following values: 0x00 – never 0x01 – any device MAY request 0x02 – Only devices in the *apsDeviceKeyPairSet* with a KeyAttribute value of PROVISIONAL_KEY MAY request.	This is set to 0x00 in networks with higher level protocols for establishing link keys. This is set to either 0x01 or 0x02 in centralized security networks.
network-KeyUpdatePeriod	0xb9	Integer	0x00000000 – 0xFFFFFFFF	The period, in minutes, of how often the network key is updated by the Trust Center. A period of 0 means the Trust Center will not periodically update the network key (it MAY still update key at other times).	This is used in the Trust Center of centralized security networks to establish the network key update period. When this time is up the Trust Center updates the network key.
network-KeyUpdateMethod	0xba	Enumeration	0x00 – 0x01	This value describes the method the Trust Center uses to update the network key. 0x00 – Broadcast using only network encryption 0x01 – Unicast using network encryption and APS encryption with a device's link key.	This is used in centralized security networks to establish the policy for updating the network key.
allowApplicationKeyRequests	0xbb	Enumeration	0x00 – 0x02	This value determines how the Trust Center SHALL handle attempts to request an application link key with a partner node. 0x00 – never 0x01 – Any device MAY request an application link key with any device (except the Trust Center) 0x02 – Only those devices listed in *applicationKeyRequestList* MAY request and receive application link keys.	This is used in centralized security networks to establish the Trust Center policy on providing Application Link keys between devices on the network. It is normally set to 0x01 allowing any device to request a link key with another device to support those applications that want to encrypt application payload.

[10] CCB 2446

Attribute	ID	Type	Range	Description	Usage
applicationKeyRequestList	0xbc	List of address pairs	Variable	This is a list of IEEE pairs of devices, which are allowed to establish application link keys between one another. The first IEEE address is the initiator, the second is the responder. If the responder address is set to 0xFFFFFFFFFFFFFFFF, then the initiator is allowed to request an application link key with any device. If the responder's address is not 0xFFFFFFFFFFFFFFFF, then it MAY also initiate an application link key request. This list is only valid if *allowApplicationkeyRequests* is set to 0x02.	This list is normally not used in centralized security networks unless the Trust Center policy restricts those devices allowed to request link keys.
allowRemoteTcPolicyChange	0xbd	Boolean	TRUE or FALSE	This policy indicates whether a node on the network that transmits a ZDO Mgmt_Permit_Join with a significance set to 1 is allowed to effect the local Trust Center's policies.	
allowVirtualDevices	0xbe	Boolean	TRUE or FALSE	This Boolean indicates whether the Trust Center is currently allowing Zigbee Direct Virtual Devices (ZVDs) to join the network. A value of TRUE means that the Trust Center is allowing such devices.	This is set to FALSE in centralized security networks that do not want to allow Zigbee Direct Virtual devices on the network.

4.7.3.1 Allowing Devices to Join using Symmetric Key

As an optional first step, the Trust Center MAY provide a means to enter in the new device and its associated Install Code into the apsDeviceKeyPairSet. This will then provide a means for the device to perform an authenticated joining. The apsDeviceKeyPairSet entry is setup as follows:

- DeviceAddress = <New Device Address>
- LinkKey = <Install Code derived key>
- Passphrase = <Install Code derived key>
- KeyAttributes = PROVISIONAL_KEY
- apsLinkKeyType = Unique
- InitialJoinAuthentication = INSTALL_CODE_KEY

If the Trust Center receives notification that a device is joining the network via the APSME-UPDATE-DEVICE.indication with the Status field set to Standard Device Unsecured Join (0x01), the following procedure SHALL be performed:

1. If allowJoins is set to FALSE, the following SHALL be done.
    a. The Trust Center SHALL proceed to the process of rejecting the join described in section 4.7.3.6. No further processing SHALL be done.

2. Search the apsDeviceKeyPairSet for an address that matches the IEEE of the joining device.
    a. If none is found and the TC Policy requireInstallCodesOrPresetPassphrase is 0x02, do the following:
        i. The TC SHALL proceed to the process of rejecting the join described in section 4.7.3.6. No further processing SHALL be done.
    b. Otherwise, add an apsDeviceKeyPairSet entry for the new device with the following properties:
        i. DeviceAddress = <New Device Address>
        ii. LinkKey = “ZigbeeAlliance09”
        iii. Passphrase = “Value of *apscWellknownPSK*”
        iv. KeyAttributes = PROVISIONAL_KEY
        v. apsLinkKeyType = Global
        vi. InitialJoinAuthentication = NO_AUTHENTICATION
3. The device has been authorized for admission to the network and the following SHALL be performed.
    a. Generate an APSME-TRANSPORT-KEY.request primitive with the following parameters.
        i. Set the DestAddress to the DeviceAddress of the APSME-UPDATE-DEVICE.indication.
        ii. Set the StandardKeyType to Standard Network Key (0x01).
        iii. Set the TransportKeyData to the Key field of the active network key entry in the nwkSecurityMaterialSet NIB attribute.

Figure 4-41 shows the process for device joins using a symmetric key.

The VerifiedFrameCounter field in the apsDeviceKeyPairSet is updated under following situations:

1. VerifiedFrameCounter is set to FALSE for all entries of apsDeviceKeyPairSet when the local device is reset. This is to indicate that the incoming frame counter is now un-reliable for the remote device and a frame counter synchronisation is required.
2. VerifiedFrameCounter is set to TRUE for the Trust Center on the joining device when the APS Encrypted Transport Key command is received.
3. VerifiedFrameCounter is set to TRUE when a confirm key frame is generated with a status of SUCCESS for a remote node after receiving a verify key.
4. VerifiedFrameCounter is set to TRUE when a confirm key frame of SUCCESS is received from the remote node.

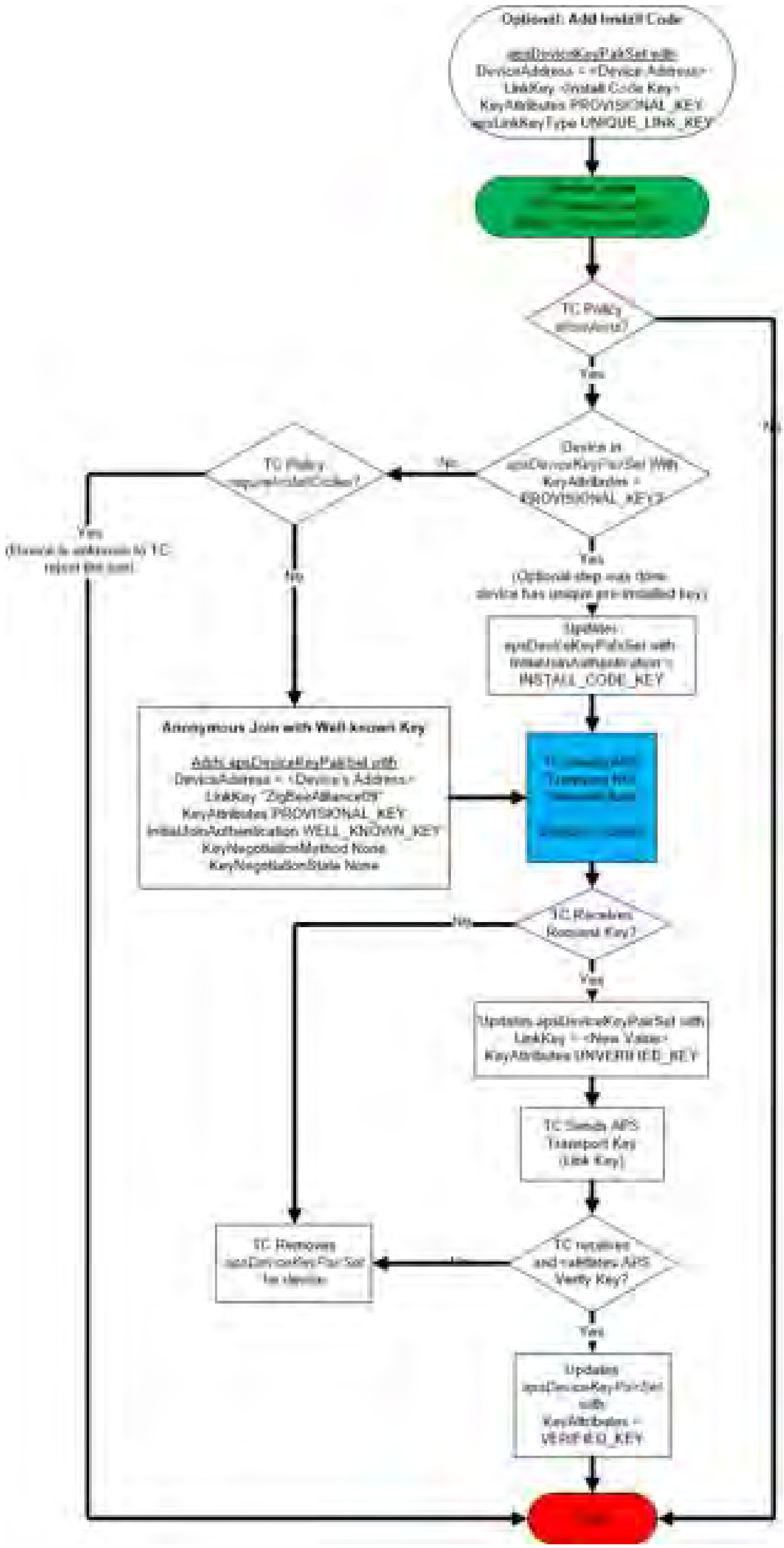


Figure 4-41. Trust Center Processing for Initial Join Using Symmetric Key

4.7.3.2 Allowing Devices to Rejoin

A device can rejoin the network at any time for various reasons. When the rejoin is not secured at the Network Layer, the device will need a copy of the active network key in order to communicate on the network. That key MUST be encrypted with a key that is unique to the device and not global. This is done to avoid exposing the network key to unauthorized devices.

For distributed networks without a Trust Center, devices SHALL NOT perform a rejoin without security at the Network Layer. Attempts to do so SHALL be rejected by routers operating in that network.

For centralized security networks, the following additional rules apply.

If the Trust Center receives notification that a device is joining the network via the APSME-UPDATE-DEVICE.indication with the Status field set to Standard Device Trust Center Rejoin (0x03) or Standard Device Secure Rejoin (0x00), the following procedure shall be performed:

1. If the Status is 0x00, consult the next higher level to determine whether the device is authorized to be on the network. If not, execute the procedure in section 4.7.3.6. Otherwise, no further action is required to allow the device to rejoin the network.
2. If the Status is 0x03, determine the Trust Center Link Key in use by the device. Lookup the device in the apsDeviceKeyPairSet Table and examine whether the entry has apsLinkKeyType set to 0x00, Unique Link Key. If it is, consult the next higher level to determine whether the device is authorized to be on the network. If no entry is found or the apsLinkKeyType is not set to 0x00, Unique Link Key, the rejoin SHALL be rejected. If the rejoin is rejected or the device is not authorized to be on the network, follow the procedure in section 4.7.3.6.

The next higher layer MAY have further rules for determining what devices are authorized to be on the network. Alternatively it MAY rely on the stack to determine this based solely on the apsDeviceKeyPairSet table data.

4.7.3.3 Allowing Devices to Join Using Key Negotation

As an optional first step, the Trust Center MAY provide a means to enter an Install Code into the apsDeviceKeyPairSet. The apsDeviceKeyPairSet entry is setup as follows:

- DeviceAddress = <New Device Address>
- LinkKey = <Install Code derived key>
- Passphrase = <Install Code derived key>
- KeyAttributes = PROVISIONAL_KEY
- apsLinkKeyType = Unique
- InitialJoinAuthentication = WELL_KNOWN_PASSPHRASE

Note that use of Key negotiation requires that parent routers are R23-compliant, supporting the APS Relay command. Routers that do not support this will require the Trust Center to use anonymous joining via the well-known key or an install code based symmetric key.

Furthermore, R23 joining devices must indicate their support for Key Negotiation by embedding a Supported Key Negotiation Methods TLV inside the Joiner Encapsulation TLV, indicating one or more asymmetric key negotiation methods.

If the Trust Center receives notification that a device is joining the network via the APSME-UPDATE-DEVICE.indication with the Status field set to Standard Device Unsecured Join and key negotiation support as described above, the following procedure SHALL be performed to negotiate a key before joining :

1. If allowJoins is set to FALSE, the following SHALL be done.
   a. The Trust Center SHALL proceed to the process of rejecting the join described in section 4.7.3.6. No further processing SHALL be done.
2. Search the apsDeviceKeyPairSet for an address that matches the IEEE of the joining device.

    a. If none is found and the TC Policy requireInstallCodesOrPresetPassphrase is 0x02, do the following:
        i. The TC SHALL proceed to the process of rejecting the join described in section 4.7.3.6. No further processing SHALL be done.
    b. Select the appropriate Key Negotiation Method based on the received Supported Key Negotiation Method TLV and the Trust Center’s supported method. Create a Selected Key Negotiation Method TLV.
    c. Otherwise, add an apsDeviceKeyPairSet entry for the new device with the following properties:
        i. DeviceAddress = <New Device Address>
        ii. LinkKey = “ZigbeeAlliance09”
        iii. Passphrase = “Value of *apscWellknownPSK*”
        iv. KeyAttributes = PROVISIONAL_KEY
        v. apsLinkKeyType = Unique
        vi. InitialJoinAuthentication = WELL_KNOWN_PASSPHRASE
        vii. KeyNegotiationMethod = (Selected Key Negotiation Method)

3. Generate a ZDO Security_Start_Key_Update_req containing the Selected Key Negotiation Method TLV
4. The Trust Center waits for a ZDO Start Key Negotiation Request and sends a ZDO Start Key Negotiation Response.
    a. If none is received, delete the apsDeviceKeyPairSet entry for that device and no further processing SHALL be done.
5. Using the Selected Key Negotiation Methods, the Passphrase in the apsDeviceKeyPairSet entry, execute the corresponding cryptographic method as described in ANNEX J. Set the entry as follows:
    a. LinkKey = <Derived Key>
    b. KeyAttributes = UNVERIFIED_KEY
    c. KeyNegotiationState = COMPLETE_KEY_NEGOTIATION
6. Wait for apsSecurityTimeout to receive and validate an APS Verify Key command
    a. If none is received, delete the apsDeviceKeyPairSet entry for that device and no further processing SHALL be done.
7. After validation of the APS Verify Key Command, set the apsDeviceKeyPairSet entry to the following:
    a. KeyAttributes = VERIFIED_KEY
    b. Frame Counter Synchronization bit in the Features & Capabilities bitmap = '1'.
8. Generate an APS Confirm Key Command to the device
9. OPTIONAL: The Trust Center MAY send or receive additional messages, encrypted at the APS layer with the newly generated key.
10. The Trust Center decides whether to allow the device on the network.
    a. If allowed, the Trust Center SHALL generate an APS Transport Key containing the network key.
    b. If not allowed, the Trust Center SHALL proceed to the process of rejecting the join described in section 4.7.3.6. No further processing SHALL be done.

Figure 4-42 shows the processing for the Trust Center of an Initial Join using Key negotiation.

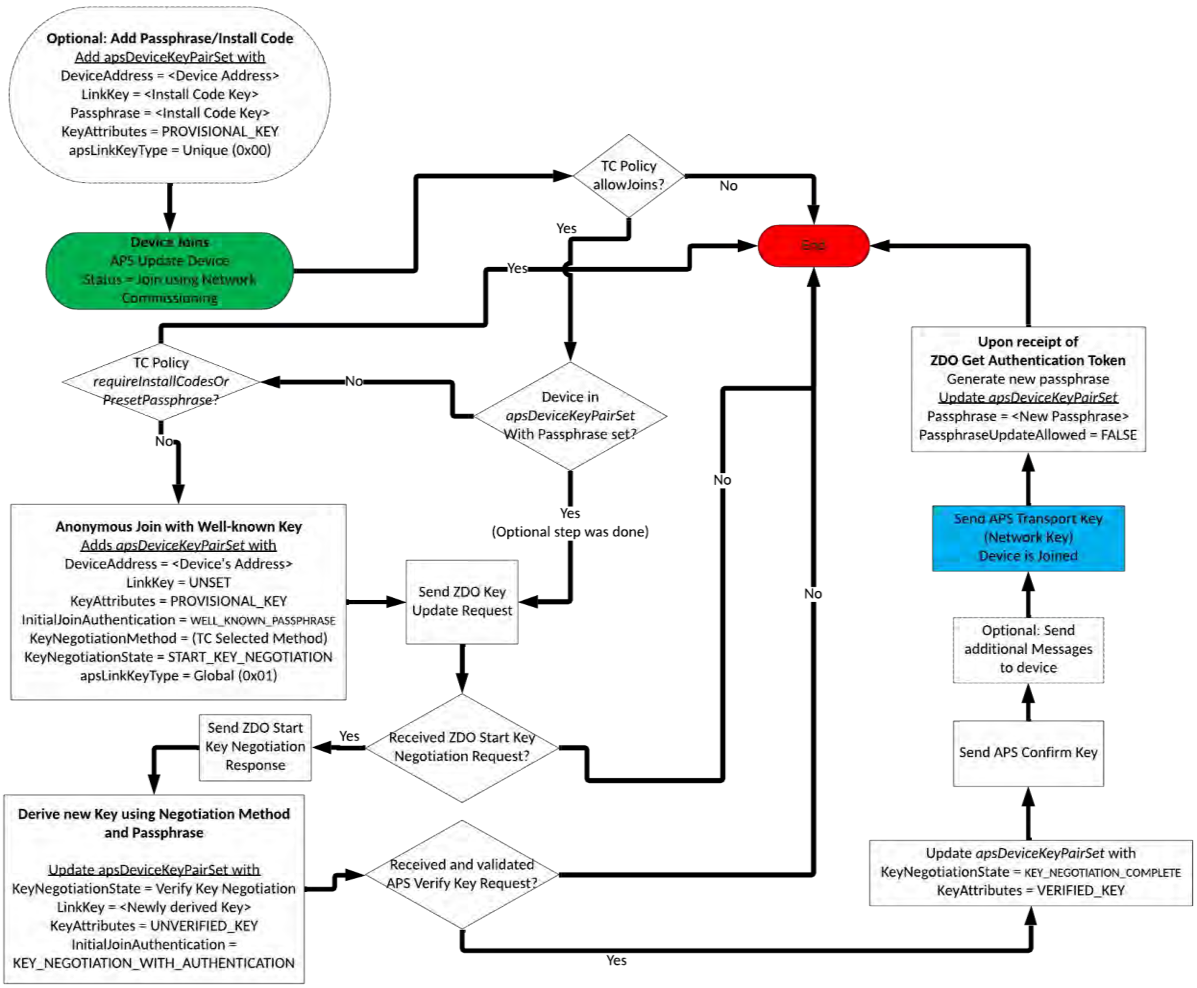


Figure 4-42. Trust Center Processing for Initial Join Using Key Negotiation

4.7.3.4 Remote Device Changing Trust Center Policy

In some networks it MAY be permissible for a joined device to request that the Trust Center allow an unjoined device to be commissioned on the network. This can be accomplished through the ZDO Mgmt_Permit_Joining_req sent to the Trust Center with the TC_Significance field set to 1. Upon receipt of this request, the following procedure SHALL be executed.

1. If allowRemoteTcPolicyChange is set to 0, then the operation SHALL be denied and the status of 0xa3 (ILLEGAL_REQUEST) passed back to the ZDO. No further processing SHALL be done.
2. If *requireInstallCodesOrPresetPassphrase* is set to 0x02, then the operation is invalid and the status of 0xaa (NOT_SUPPORTED) SHALL be passed back to the ZDO. No further processing SHALL be done.
3. The operation is allowed by the Trust Center and a status of 0x00 (SUCCESS) SHALL be passed back to the ZDO.

When a Trust Center receives a Mgmt_Permit_joining_req where the allowRemoteTcPolicyChange=FALSE, the Trust Center MAY broadcast a Mgmt_Permit_joining_req with permitDuration=0 to close the network and prevent it from advertising that new devices are being accepted.

When the new device requests to join the network the trust center will still process the joining device as described in section 4.7.3.1.

4.7.3.5 Determining the Link Key for Encryption or Decryption by the Trust Center

If the Trust Center has determined that a message SHALL be sent with APS encryption or has been received and SHALL be decrypted, it SHALL determine what link key to use for the operation. The Trust Center SHALL examine the IEEE address of the destination (if encrypting) or source (if decrypting) and search the *apsDeviceKeyPairSet* for a matching address entry. If a match is found, it will use the associated link key to APS encrypt or decrypt the message.

If no matching entry is found then no link key is defined and processing of the message SHALL be stopped. The message will not be sent or received because there is no link key that can be used.

See sections 4.4.1.1 and 4.4.1.2 for outgoing and frame processing respectively.

4.7.3.6 Rejecting the Join or Rejoin

A join or rejoin is processed via an APSME-UPDATE-DEVICE.indication. Following the decision to reject a join or rejoin the following SHALL be done by the Trust Center.

1. If the Status of APSME-UPDATE-DEVICE.indication was Standard Device Unsecured Join (0x01) or Standard Device Trust Center Rejoin (0x03), the following SHALL be done.
    a. The joining or rejoining device does not have the current network key and will be left to timeout.
    b. No further processing SHALL be done.
2. If the Status of the APSME-UPDATE-DEVICE.indication was Standard Device Secured Rejoin (0x00), the following SHALL be done.
    a. Follow the procedure in section 4.7.3.7.

4.7.3.7 Removing Devices

The Trust Center has the ability to remove devices in the network via the APS Remove Device command. This message can be used to force well-behaved devices to leave the network. This is useful if the Trust Center determines after they have joined that they are not on the correct network or that the device is unable to communicate in a required application specific way.

It is important to note that this is not a secure means of removing a device. Once a malicious device has the current network key the only way to force it off the network is to distribute a new network key in a manner that prevents the malicious device from obtaining the new key. See section 4.7.3.12.

4.7.3.8 Processing Trust Center Link Key Requests

The Trust Center link key is a crucial element in joining the network when a preconfigured key is in place, or when a device attempts to rejoin after a missed network key update. It is also the means by which application keys are established with other devices on the network.

Devices are required to update their Trust Center link key after joining if key negotiation was not used during joining. One of the main mechanisms is the Trust Center Link Key Request using the APS Command Request Key. However, if a device has negotiated a link key using a different mechanism than APS Command Request Key then the Trust Center SHALL reject attempts to update using APS Command Request Key. More secure mechanisms for updating the link key SHALL NOT be overridden by a Trust Center Link Key request.

The process in Zigbee for transporting a new link key to the device requires the previous link key as an authentication mechanism. In addition it uses APS commands which do not have support for APS retries. As a result it is possible for devices to get out of sync with regard to the Trust Center link key currently in use. To avoid this risk the Trust Center MAY decide to prohibit requests for new trust center link keys when one is already in place.

The following describes the process when the Trust Center is notified of an APS Request key via the APSME-REQUEST-KEY.indication with the RequestKeyType set to 0x04 (Trust Center Link Key):

1. If the APS Command Request Key message is not APS encrypted, the device SHALL drop the message and no further processing SHALL be done.
2. The device SHALL verify the key used to encrypt the APS command. If the SrcAddress of the APSME-REQUEST-KEY.indication primitive does not equal the value of the DeviceAddress of the corresponding apsDeviceKeyPairSet entry used to decrypt the message, the message shall be dropped and no further processing SHALL be done.
3. The Trust Center SHALL examine the KeyNegotiationMethod of the apsDeviceKeyPairSet entry for the corresponding device and if the value is NOT 0x00 then the request SHALL be silently dropped.
4. If the RequestKeyType is set to 0x04, Trust Center Link Key, the following SHALL be performed:
    a. If *allowTrustCenterLinkKeyRequests* is 0, then no further processing SHALL be done. The request is silently rejected.
    b. If *allowTrustCenterLinkKeyRequests* is 1, then the following is performed:
        i. Follow the procedure in section 4.7.3.8.1.
    c. If allowTrustCenterLinkKeyRequests is 2, do the following.
        i. Find an entry in the apsDeviceKeyPairSet of the AIB where the DeviceAddress of the entry matches the PartnerAddress of the APSME-REQUEST-KEY.indication primitive, and the KeyAttributes has a value of PROVISIONAL_KEY (0x00). If no entry can be found matching those criteria, then the request shall be silently dropped and no further processing SHALL be done.
        ii. Otherwise, follow the procedure in section 4.7.3.8.1.

4.7.3.8.1 Procedure for Generating and Sending a new Trust Center Link Key

This procedure takes an IEEE address DeviceAddress.

1. Create a new 128-bit key, KeyValue. This MAY be done using a random number generator, or programmatically using an algorithm.
2. Create a new entry in the apsDeviceKeyPairSet.
    a. Set the DeviceAddress of the entry to the DeviceAddress passed to this procedure.
    b. Set the LinkKey value of the entry to the KeyValue previously generated in this procedure.
    c. Set the KeyAttributes of the entry to UNVERIFIED_KEY (0x01).
    d. Set the ApsLinkKeyType of the entry to Unique Link Key (0x00).
3. If there is no space in the apsDeviceKeyPairSet attribute then processing SHALL fail and no further steps are executed in this procedure.
4. Issue an APSME-TRANSPORT-KEY.request primitive with the DestAddress set to the DeviceAddress, the StandardKeyType set to 0x04 (Trust Center Link Key), and the TransportKeyData set to the KeyValue.

4.7.3.9 Alternate methods of Updating the Trust Center Link Key

Updating the Trust Center link key using APS request key or unsolicited transport key messages is problematic due to synchronization issues. Neither side knows which key the other side is using and future attempts to update the key require knowledge of the current key.

An alternate mutual authentication protocol SHALL have all of the following properties:

1. The protocol SHALL use one or more shared secrets that are not transmitted over the air during the protocol negotiation.
2. The protocol SHALL allow both sides to inject random data in the key generation to prevent one device from controlling the result of the key generation.

3. The protocol SHALL not require knowledge of a previously generated Trust Center link key in order to generate a new one.

Both CBKE (Certificate Based Key-Exchange) and DLK (Dynamic Link Key) have all of these properties.

4.7.3.10 Processing Application Link Key Requests

Devices MAY use the Trust Center to establish application link keys with one another. Those devices can leverage the secure communications channel they have established with the Trust Center in order to establish secure communications with other devices. The Trust Center policy dictates whether or not it will answer application link key requests. Trust Center SHALL only allow application link key requests it receives that are encrypted with the device's Trust Center link key. Any application link key request that is not APS encrypted shall be dropped. In addition, if the Trust Center does not have a link key in *apsDeviceKeyPairSet* for the responder device listed in the APS Request Key Command, it SHALL drop the request. The purpose of the using the Trust Center to establish an application link key is leverage the trust each device has with the Trust Center (through their Trust Center Link Key).

The Trust Center SHALL ignore any requests made to establish application link keys with itself. Zigbee provides no protocol mechanism to differentiate whether a Trust Center link key or an application link key was used to encrypt a message. Therefore a device cannot determine what key to use when decrypting the message.

It is worth noting that devices are not required to use the Trust Center to establish application link keys, and that some application profiles allow devices to establish link keys without the trust center. The application profile in use by the device MAY require that the Trust Center be utilized to do this.

Application link key requests are initiated by the requesting device MAY occur at any time. Therefore the Trust Center SHALL NOT change its handling of those requests based on whether it is currently operating in commissioning or operational mode.

Upon receipt of an APSME-REQUEST-KEY.indication with the RequestKeyType set to 0x02 (Application Link Key) the following SHALL be performed:

1. If the PartnerAddress of the primitive is equal to the apsTrustCenterAddress of the AIB, the request SHALL be dropped and no further processing SHALL be done.
2. If the Trust Center policy of allowApplicationLinkKeyRequests is 0x00, then the request SHALL be dropped and no further processing SHALL be done.
3. If the Trust Center policy of allowApplicationLinkKeyRequests is 0x01, then the Trust Center SHALL do the following.
    a. Run the procedure in section 4.7.3.10.1 using SrcAddress from the primitive as the InitiatorAddress in the procedure, and PartnerAddress from the primitive as the ResponderAddress in the procedure.
    b. No further processing SHALL be done.
4. If the Trust Center policy of allowApplicationLinkKeyRequests is 0x02, then the following SHALL be performed.
    a. Find an entry in the allowApplicationKeyRequestList where the SrcAddress of the primitive matches the Initiator Address of the entry, and the PartnerAddress of the primitive matches the Responder Address of the entry.
    b. If no entry is found, then the request SHALL be dropped and no further processing SHALL be done.
    c. If an entry is found, follow the procedure in section 4.7.3.10.1.

4.7.3.10.1 Procedure for Generating and Sending Application Link Keys

This procedure takes two IEEE addresses, InitiatorAddress and ResponderAddress.

1. The Trust Center SHALL generate a random 128-bit key KeyValue for the application link key.
2. It SHALL issue an APSME-TRANSPORT-KEY.request with the StandardKeyType set to 0x03, Application Link Key, the TransportKeyData set to KeyValue, and the DestAddress set to InitiatorAddress.

3. It SHALL issue a second APSME-TRANSPORT-KEY.request with the StandardKeyType set to 0x03, Application Link Key, the TransportKeyData set to KeyValue, and the DestAddress set to ResponderAddress.

4.7.3.11 Key Lifetime

How long a network key or trust-center link key remains in use is up to the trust center. The longer a key is in use the more chance there is of it becoming compromised. On the other hand, updating a key too often adds management overhead and increases the risk that problems during key transmission will disrupt the network. A balance SHALL be struck between the needs of security and the temporary disruption a new key can cause.

4.7.3.11.1 Link Key Lifetime

- It is advisable that the trust center have a policy for link keys to be changed periodically. This is can be difficult for sleepy end devices, which SHALL check with the trust center periodically to receive any newly-available key.
- It is recommended that old, unused link keys be deleted from the Trust Center to prevent them from being used. This requires that devices periodically communicate with the Trust Center using APS security to allow it to keep track of usage of the keys.
- Often a link key is used to initially join the network and thus it is uncertain how long the key MAY have been in use before joining the network. Preconfigured link keys MAY be extremely long lived and thus increases the need to update the link key as soon as the device joins the network.
- Link keys that are established using higher level protocols are not updated based on trust center policies but on the higher level application policies.

4.7.3.11.2 Network Key Lifetime

The trust center SHALL periodically distribute and then switch to a new network key. There are two main reasons for doing this:

1. An update and switch resets the outgoing NWK frame counter of all devices on the network. This lengthens the life of the network, since once the frame counter of a device gets to all 0xFFFFFFFF it cannot send network encrypted traffic.
2. It reduces the risk of a network key being compromised through attacks.

If a trust center detects that the frame counter for any device in its neighbor table is greater than 0x80000000 it SHOULD update the network key.

Trust centers SHOULD update the network key at least once per year. It is not recommended to update the network key more than once every 30 days except when required by the application or profile.

Trust centers that do not have a real time clock or other means of tracking time are recommended to perform a network key update when their outgoing frame counter reaches 0x40000000.

4.7.3.12 Updating the Network Key

Updating the Network key is one of the core responsibilities of the Trust Center. It helps to insure that a key does not remain in use for too long and thus is not too susceptible to compromise.

4.7.3.12.1 Period of Updates

The network key SHALL be updated periodically. How often an update is sent out is based on the *nwkKeyTrustCenterUpdatePeriod*.

4.7.3.12.2 Sleepy Devices

Sleepy devices MAY miss many network events, such as a channel change, PAN ID change, or a parent that has left. Sleepy devices MAY NOT be awake at the point when the Trust Center is updating the network key, regardless of whether the key is broadcast or unicast. If the sleeping device happens to poll within nwkcTransactionPersistenceTime for a unicast key update, or nwkcBroadcastDeliveryTime for a broadcast key update, the update message SHALL be delivered. Otherwise the delivery of the key update to the sleepy device will timeout and the sleepy device will not receive the update.

The sleepy device SHOULD consider the network key update another one of those events and will need to handle that case when waking up. A child that sends a message to its parent and receives a MAC ACK but no response at the application layer MAY have missed a key update and therefore SHOULD try to perform a rejoin. If the parent has switched to the newer key, the sleeping device SHALL perform a trust center rejoin.

4.7.3.12.3 Broadcast Network Key Updates

Broadcast key updates are the simplest mechanism for distributing new network keys. The new network key is broadcast using the existing network key to encrypt it. There is no way to exclude a device that has the current network key from this kind of key update.

4.7.3.12.4 Unicast Network Key Updates

A more secure way of sending out network key updates is by using unicast messages encrypted with each device's link key. This requires that all authorized devices on the network have a link key so that the Trust Center can individually update them in a secure manner. A Trust Center that wishes to securely remove a previously authorized device SHOULD use this mechanism as it can be used to exclude a device from the network.

If this unicast method is used by the trust center, it is required that the Trust Center maintain a list of all the routers on the network and send key updates to only those devices. Sleepy devices are unlikely to be awake when the Trust Center decides to change the network key. Sending to only routers also reduces the amount of network traffic that the Trust Center has to generate in order to update the network.

4.7.3.12.5 Key Switch

Regardless of the mechanism used to perform a key update (broadcast or unicast), it is required that the APS key switch command be broadcast. Devices will implicitly switch the network key when they hear another device using the newer key. This mechanism insures that even if the device did not receive the formal key switch, it will start using the new key.

A device can determine if the new network key is actively being used by examining the key sequence number in the NWK auxiliary header of packets it receives. If it receives a message that passes decryption using the new key sequence number then it SHALL switch to using the newer network key and stop sending message encrypted using the old network key.

4.7.3.12.6 Old Network Keys

A network key update and switch does not preclude the use of the previous network key. A device is allowed to accept messages encrypted using the last network key, as this insures a smooth transition to the new key. A device SHALL never send messages using the old key.

To completely deprecate a key's use, the Trust Center will have to perform an update and switch twice. If using a broadcast key update, the Trust Center SHOULD make sure that both the key update and the key switch broadcasts have completely expired before sending a second set to update and switch.

4.7.4 Trust Center Swap-out Overview

The Trust Center is a key role in a centralized security network. As such it is important to have a process to securely replace a failed trust center. The general process is that a subset of the security material and NIB values of the Trust Center will be backed up off-network. Once the old Trust Center has been removed the new Trust Center can be brought online with the Trust Center backup data.

The application layer SHOULD employ mechanisms to detect the loss of the Trust Center. It is expected that the application layer keepalive mechanism will be used for this purpose. The details of the application layer keepalive are outside the scope of this specification. Upon the detection of one or more failed keepalives the application will trigger an NLME-JOIN.request with RejoinNetwork = 2 (network rejoin procedure) and SecureRejoin = FALSE.

Often the loss of a Trust Center is a transitory issue and connectivity will return to normal. However, it is possible the existing trust center has enacted a network wide change such as changing the network key, radio channel, or PAN ID. A Network Rejoin will fix this issue. The least likely scenario, but an important one to handle, is that the trust center has been replaced and its identity and security material have changed.

There are two methods of Trust Center Swap-out: a less secure method that supports networks with legacy devices, and a more secure method that only supports devices compliant with Revision 23 of the specification and later.

For the first method that supports legacy devices, a set of security and network information is backed up off-device, including active security keys, and all of this information is restored onto a new Trust Center when required. Because extracting security information off-device can pose a security risk, Trust Center Swap-out support for networks with legacy devices is an optional feature.

For the second method that only supports new devices, security and network information is also backed up but security keys will be run through a hash function before they are extracted off-device, thus preventing a passive network attack in the case where the backup data set is compromised. This method also changes the EUI64 of the Trust Center when it is swapped out, so devices will be able to differentiate between the swap-out methods.

Nodes that perform a Trust Center rejoin and notice that the Trust Center EUI64 has changed will obtain a new network key by creating the hashed variant of the current Trust Center link key to decrypt the Transport Key message. After any application layer verification of the new Trust Center, the nodes will now be bound to the new Trust Center. Even if the old Trust Center returns, nodes that have made the switch will not go back to the old Trust Center.

After being bound to the new Trust Center the device MUST update its existing link key and a new backup for that key SHALL be created.

All Zigbee devices are required to support Trust Center Swap-out.

4.7.4.1 Trust Center Backup

4.7.4.1.1 Networks with Legacy Devices

A Legacy device for Trust Center Backup is one running Revision 22 or earlier.

4.7.4.1.1.1 ***NIB / AIB Values***

In order to replace the existing Trust Center in a network with legacy devices, the new Trust Center must use the same set of network and security information as the original Trust Center.

The following are the values needed for the backup:

4.7.4.1.1.1.1 NIB Values

Value
nwkExtendedPanId
nwkSecurityMaterialSet
nwkActiveKeySeqNumber
nwkPANId
nwkIeeeAddress
Network Channel (via nwkMacInterfaceTable)

4.7.4.1.1.1.2 AIB Values

Value
apsBindingTable
apsDeviceKeyPairSet (all elements of the entry)
apsTrustCenterAddress
trustCenterPolicies

4.7.4.1.1.2 ***Period of Backup***

A backup of the Trust Center is required each time any of the values in section 4.7.4.1.1.1 are modified on the Trust Center.

4.7.4.1.1.3 ***Replacing the Trust Center***

The following steps are to replace an old Trust Center in a network with legacy devices.

1. Backup of NIB and AIB values as described in section 4.7.4.1.1.1.1 and section 4.7.4.1.1.1.2. The period of backup is described in section 4.7.4.1.1.2.
2. New Trust Center Device is provisioned and installed
    a. New Trust Center NIB / AIB values are set to the values restored from a backup
    b. New Trust Center calls NLME-NETWORK-FORMATION.req

NOTE: The new Trust Center will use the IEEE Address of the old Trust Center. It will NOT use its own IEEE address.

For networks with legacy devices, the replacement of the Trust Center should be seamless, i.e. devices on the network should not need to perform any new action to reestablish connection with the new Trust Center. The Trust Center MAY perform additional actions at this point such as rotating the network key, but any further actions would be application specific.

4.7.4.1.2 Networks with Revision 23 Devices and later

In order to replace the existing Trust Center in a network, a subset of the material contained in the AIB and NIB is required. It is important to note that this backup never requires that the active encryption keys to be stored. The application may choose to backup additional application layer data as it sees fit, but it SHALL only backup the apsDeviceKeyPairSet items as noted in Table 4-43.

The application may choose to backup additional application layer data as it sees fit, but it SHALL only backup the apsDeviceKeyPairSet items as noted in Table 4-43.

The following are the values needed for the backup:

4.7.4.1.2.1 ***NIB Values***

Value
nwkExtendedPanId

4.7.4.1.2.2 ***AIB Values***

It is important to note that the neither the current nor alternate network key is backed up. This avoids compromising security material that is actively in use by the network.

4.7.4.1.2.2.1 AIB Values

Value	Mandatory
apsDeviceKeyPairSet (partial)	Mandatory
trustCenterPolicies	Optional

Each entry in the apsDeviceKeyPairSet SHALL be backed up but not all elements contained within an entry are backed up as shown in Table 4-43. This is done to avoid backing up actively used security material.

Table 4-43. Items to Back Up from the apsDeviceKeyPairSet

Value	Backed Up
DeviceAddress	Yes
KeyAttributes	Yes
LinkKey	No
OutgoingFrameCounter	No
InitialJoinAuthentication	Yes
KeyNegotiationMethod	Yes
KeyNegotiationState	No
Passphrase	Yes (if supported)
Timeout	No
PassphraseUpdateAllowed	No
TrustCenterSwapOutLinkKey	Yes

4.7.4.1.2.3 Link Key Special Handling (TrustCenterSwapOutLinkKey)

In order to protect the security of the network, a hash on the TC link key will be performed and that will be the key stored externally. It is highly recommended that the actual link key used for operational networks never be transported out of the Trust Center. Using this method, if the backup data for the TC is compromised then it cannot be used to passively compromise existing Zigbee network communications.

4.7.4.1.2.4 Process for Creating the TrustCenterSwapOutLinkKey

The new hashed version of the Link key shall be created by performing a 128-bit AES-MMO hash on the 128-bit key data of the LinkKey element of the entry from the apsDeviceKeyPairSet (See section B.1.3 for details of the hashing algorithm.) Table 4-44 defines the test vector for the hash.

Table 4-44. Test Vector for Hash of the Trust Center Link Key

Trust Center Link Key	C0C1C2C3C4C5C6C7C8C9CACBCCCDCECF
Hashed Trust Center Link Key	A7977E88BC0B61E8210827109A228F2D

The AIB apsDeviceKeyPairSet stores the value of key used for Trust Center swap out in the TrustCenterSwapOut-LinkKey value. The TrustCenterSwapOutLinkKey value SHALL be replaced with the hashed version of the Trust Center Link Key whenever the Trust Center Link Key is updated.

4.7.4.1.2.5 ***Period of Backup***

A backup of the Trust Center is only needed when a device changes its Trust Center link key. This could occur when a device is added to the network, a device is removed from the network, or when the device updates the trust center link key.

Changes to the Network Key do not require a new backup.

4.7.4.1.2.6 **Replacing the Trust Center**

The following are the steps to replace the old trust center.

1. Backup of NIB and AIB values as described previously.
2. New Trust Center Device is provisioned and installed.
   a. The New Trust Center will use its own EUI64 as the *apsTrustCenterAddress.*
   b. New Trust Center will randomly generate a new short PAN ID that is different than the old PAN ID.
   c. New Trust Center will randomly generate a new Network Key and sequence number.
   d. New Trust Center calls NLME-NETWORK-FORMATION.req.
3. Import the AIB values that were backed up with the additional special handling.
   a. Set each associated LinkKey value to the value of the appropriate TrustCenterSwapOutLinkKey.
   b. Recalculate the TrustCenterSwapOutLinkKey from the LinkKey as described in section 4.7.4.1.2.4.
4. Application detects the loss of the trust center.
5. Application will trigger nodes to rejoin to the new Trust Center using a Trust Center Rejoin operation.
   a. This will be an NLME-JOIN.req with RejoinNetwork parameter set to 2 (network rejoin procedure) and SecurityEnable set to FALSE.
   b. Nodes will scan for the current extended PAN ID but short PAN ID will have changed.
6. The new Trust Center will send the new network key to the device via the APS Transport Key Command.
   a. The message SHALL be encrypted at the APS layer with the Trust Center's current LinkKey for that device.
      i. For the device, this will be the value of the TrustCenterSwapOutLinkKey from the device's own entry in its apsDeviceKeyPairSet AIB item.
   b. The message SHALL use the new Trust Center's EUI64 as the Source Address in the APS Command frame.
   c. The message SHALL set the Extended Nonce of the APS Security Auxiliary Header to TRUE and set the Source Address in the Auxiliary Header to the new Trust Center's EUI64.
7. The device will notify the application via the APSME-TRANSPORT-KEY.indication primitive and SHALL update the value used for LinkKey in the apsDeviceKeyPairSet entry. See section 4.4.2.2.
8. The device will perform a key update mechanism according to its local supported mechanisms and the supported mechanisms of the Trust Center. It is important to note that an update to the Trust Center Link Key for a device is required after rejoining to the new Trust Center. This ensures that the existing backup of the link key is not used as an active encryption key. Once the link key has been updated a hash of that key is backed up as previously described.

4.7.4.1.2.7 **Node Behavior**

In order for the node to detect the loss of the Trust Center there must be an application layer keepalive to the Trust Center. The definition of that keepalive is outside the scope of this specification.

When the application layer keepalive has failed there are several different possible reasons. They are listed below from most likely to least likely.

1. The Trust Center is temporarily unavailable.
2. The device missed a network key update by the Trust Center and is using an old network key.
3. The Trust Center changed channels or PAN IDs to avoid congestion.
4. The Trust Center has been swapped out.

To fix any of these issues the application can trigger a rejoin operation via the NLME-JOIN.req with RejoinNetwork parameter set to 2.

When the Node is performing a Trust Center rejoin and receives an APS Command with APS encryption it SHALL do the following additional behavior.

1. Execute the Security Processing of Incoming [APS] Frames as described in section 4.4.1.2.
2. If security processing fails, execute the Security Processing a second time using the TrustCenterSwapOutLinkKey as the link key for decryption. The frame counter check SHALL not be performed.
    a. If security processing succeeds, then a Trust Center Swap-out has occurred. The device SHALL execute the procedure described in section 4.7.4.1.2.9.
    b. The join has succeeded and the NLME-JOIN.confirm SHALL be issued.
    c. The apsDeviceKeyPairSet entry associated with the apsTrustCenterAddress SHALL set the KeyAttributes element to PROVISIONAL_KEY.
3. If security processing fails a second time, continue executing the NLME-JOIN.req.

4.7.4.1.2.8 AIB Update Due to New Trust Center

If the node detects a Trust Center Swap-out has occurred, it SHALL do the following.

1. Find the entry in the apsDeviceKeyPairSet of the AIB corresponding to the existing apsTrustCenterAddress.
    a. Copy the value of the TrustCenterSwapOutLinkKey to the LinkKey value.
    b. Recalculate the TrustCenterSwapOutLinkKey based on the new LinkKey value
    c. Update the DeviceAddress of the entry to the value of the SourceAddress in the APS Transport Key Command.
2. Update the apsTrustCenterAddress in its AIB with the value of the SourceAddress in the APS Transport Key Command.

4.7.4.1.2.9 Notification to the Application

The application will be notified of the change in Trust Center via the APSME-TRANSPORT-KEY.indication primitive. The APSME-TRANSPORT-KEY.indication will contain a SrcAddress Parameter that DOES NOT match the current AIB apsTrustCenterAddress field.

The application SHOULD accept the new Trust Center. However, the application MAY perform additional checks based on its own security requirements to authenticate the new trust center.

If the new Trust Center is accepted, the application MUST perform an APSME-SET.req to update the AIB apsTrustCenterAddress with the new value received in the SrcAddress of the APSME-TRANSPORT-KEY.indication primitive. The application MUST also update the current LinkKey value for the Trust Center entry of the apsDeviceKeyPairSet AIB item. The value of the TrustCenterSwapOutLinkKey is copied to the LinkKey value using the APSME-SET.req.

The stack will then automatically update the TrustCenterSwapOutLinkKey for that entry based on the new LinkKey value, as described in section 4.7.4.1.2.4.

4.7.4.1.2.10 Additional Behavior after Trust Center Swap-out

The application may require additional behavior from the node once a Trust Center Swap-out has occurred and the device has successfully rejoined to the network. It may need to query the node for additional data that was not backed up, or it may choose to do additional security updates such as changing the Trust Center Link Key from the value that was in the backup.

The application will have to perform additional changes to other application data. For example, bindings to the old Trust Center would have to be updated to the new Trust Center's EUI64 and MAY require a re-discovery to update endpoint information.

The application SHOULD not perform any APS encrypted messaging until the link key has been updated. This means performing a link key update using the application’s designated mechanism. This will change the value that was used by the new Trust Center to perform the swap-out (apsDevicekeyPairSet.TrustCenterSwapOutLinkKey).

4.7.4.1.3 Networks with a Mix of Revision 23 and Earlier Devices

It is possible to support Trust Center Swap-out with a mix of devices from Revision 23 and later, and those that are running a stack of Revision 22 and earlier. For those devices that support Revision 23 a Trust Center can determine that support and avoid backing up the active encryption key and only back-up a hash of the key as described earlier. For those devices that do not support Revision 23 the Trust Center can backup the link key as described in the prior section.

This would mean a backup of a heterogeneous network would contain a mix of active encryption keys for devices which implement legacy revisions of this specification and only hashes of the active key for R23 and profiles which support it. Even in this case there is still benefit as it reduces the active security material that must be backed up and thus is vulnerable to attack.

4.8 Security Operations in Distributed Security Networks

In distributed security networks, there is not a single trust center in control of the network. Each router can act as a parent and transport keys to joining devices. In addition, if a device already has a network key from an out of band installation method or commissioning, the device is accepted into the network without any trust center authorization.

4.8.1 Trust Center Address

In distributed security networks the trust center address is 0xFFFFFFFFFFFFFFFF. This address is used in transport key commands as the source address to indicate the network is in a distributed security model.

4.8.2 Network Key Updates

Network key updates are not done in distributed security networks.

4.8.3 Link Keys

Link keys are only used to APS encrypt transport key commands during joining in distributed security networks. The key type stored internally SHALL be 0x01 (Global Link Key).

4.8.4 Application Link Keys

Devices MAY require use of application link keys for APS data. In a distributed security network the partner devices SHALL use a higher level protocol to establish the application link key without the trust center involvement or permissions.

4.8.5 Requesting Keys

There is no facility to process or answer APSME-REQUEST-KEY primitives. All APS Command Request Key frames shall be dropped and no further processing SHALL be done.

4.9 Device Operations

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

4.9.1 Joining Device Policy Values

A joining device MAY have a set of policy values enumerated in Table 4-45. However, it normally sets these policy values upon joining based on if the network is a centralized or distributed security model. All devices SHALL support joining either network and adapting their security policies accordingly unless their application profile mandates joining only one type of network.

Table 4-45. Joining Device Policy Values

Name	Type	Range	Description	Usage
requestNewTrustCenterLinkKey	Boolean	TRUE or FALSE	This Boolean indicates whether the device will request a new Trust Center Link key after joining. A value of TRUE means the device SHALL send an APS request key command to the Trust Center with RequestKeyType 0x04. If the request is not answered requestLinkKeyTimeout seconds then the device will leave the network. A value of FALSE means the device will not request a new link key.	This is set to TRUE in centralized security networks to ensure devices have a trust center link key for rejoining or key updates. Note this value is set to FALSE in a distributed security network.
requestLinkKeyTimeout	Integer	0 – 10	This integer indicates the maximum time in seconds that a device will wait for a response to a request for a Trust Center link key.	This is ignored in a distributed security network.
acceptNewUnsolicitedApplicationLinkKey	Boolean	TRUE or FALSE	This Boolean indicates whether the device will accept a new unsolicited application link key sent to it by the Trust Center.	
requireLinkKeyEncryptionForApsTransportKey	Boolean	TRUE or FALSE	This indicates whether or not the device will require that the APS Transport Key command SHALL be APS encrypted with the device's unique Trust Center Link Key.	By default this is FALSE.

4.9.2 Trust Center Address

A device will not know the address of the Trust Center prior to joining. The *apsTrustCenterAddress* in the AIB SHALL be initially set to 0x0000000000000000. Upon joining a device SHALL receive an APS Transport key and the source address SHALL indicate the address of the trust center. The *apsTrustCenterAddress* in the AIB will be set to the address in the received packet.

A value of 0xFFFFFFFFFFFFFFFF for the *apsTrustCenterAddress* in the AIB indicates a distributed security network and the device settings SHOULD be adjusted accordingly.

See section 4.4.1.5 for a description of when and how the trust center address of APS commands are validated.

4.9.3 Trust Center Link Keys

All devices implementing this specification SHALL support one or more methods to update their link key. At a minimum, they SHALL support the Key Request Update Mechanism. They MAY support others. Devices SHALL indicate their support in the Supported Key Negotiation Methods Global TLV.

All devices in a centralized security network SHALL obtain an updated Trust Center link key when they first join the network and the Trust Center supports this behavior. An updated trust center link key protects the device from compromise if the original joining key is discovered. The application MAY utilize a key establishment algorithm if one is available. If such an algorithm is not available, the Request Key services of the APSME SHALL be used.

Prior to Revision 21 of this specification, there was not an interoperable mechanism to update the link key so. Therefore a Trust Center operating on a prior Revision is not assumed to have support for this behavior. Determining the Trust Center Revision can be done using the Server Mask and the ZDO Node Descriptor Request. Initiation of this process is done by the higher application.

Once the device has obtained an updated Trust Center link key it SHALL ignore any APS commands from the Trust Center that are not encrypted with that key.

4.9.4 Receiving New Link Keys

It is possible a device's security policy MAY restrict application link keys sent to it by the trust center for use with another partner device. This could be because the device wishes to control which other devices it shares link keys with, or because it uses some other mechanism to establish application link keys with devices besides the trust center.

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 SHALL be established through the Certificate Based Key Exchange protocol.

If the devices 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.

4.9.5 Requesting a Link Key

If both the joining device and trust center support a key negotiation mechanism they SHALL use that to update the link key and SHALL NOT use this Request Key method to update the Link Key. Otherwise the joining device SHALL update its key via the request key method described below.

A device SHALL attempt to update its trust center link key as part of its initial joining operations in a centralized security network. Trust Centers prior to the Revision 21 version of this specification did not support updating trust center link keys via the APSME request key method. Determination of whether the trust center supports this behavior is left up to the higher level application. The application MAY use either the APSME Request Key facilities or an alternative key establishment protocol.

If the device is requesting a trust center link key using the APSME, it SHALL start a timer after sending the initial request. Once the timer has reached *requestLinkKeyTimeout*, the device SHALL no longer accept a transport key message containing a new Trust Center link key unless the device initiates a new request.

If the device is requesting an application link key and acceptNewUnsolicitedApplicationLinkKey is set to FALSE, it SHALL start a timer after sending the initial request. Once the timer has reached requestLinkKeyTimeout the device SHALL no longer accept a transport key message containing a new application link key unless it initiates a new request.

A device that did not request a new application link key and has acceptNewUnsolicitedApplicationLinkKey set to FALSE SHALL silently drop the APS Transport Key Command for an application link key. It SHALL NOT process the command.

4.9.6 Negotiating a Trust Center Link Key

A device with support for negotiating a link key SHALL prefer using that mechanism to establish a new Trust Center Link Key over the Request Key method. If the Trust Center does not support negotiation a link key the device can fall back to using the Request Key Mechanism.

A device uses the APSME-KEY-NEGOTIATION.request primitive to initiate the process either before joining the Zigbee network, or after joining the network. It is preferred to do negotiate a link key before joining a Zigbee network. However, when a router without APS Relay support is between the Trust Center and the joiner it is necessary for the device to complete the joining operation first. After completing the joining operation and receiving the network key the device can learn the capabilities of the Trust Center. If both the device and Trust Center support negotiating a link key SHALL be the preferred mechanism.

Application defined key negotiation mechanisms MAY be used as an alternative to the APSME-KEY-NEGOTIATION primitives. This is outside the scope of this specification.

4.9.7 Updating the Trust Center Passphrase after Initially Joining a Network

After a device first joins the network and performs Key Negotiation with the Trust Center, it SHALL update its initial passphrase. This passphrase is stored in the Passphrase field of the AIB apsDeviceKeyPairSet entry associated with the Trust Center's EUI64. It is only required to do this behavior if the Trust Center Link Key was updated using Key Negotiation; other mechanisms (such as Request Key) do not require this behavior to be performed.

A Trust Center Link Key passphrase update is required if the device negotiates a Trust Center Link Key before joining the network, or if it joins and negotiates a new link key after joining the network. See Figure 1-5, Joining in Revision 23 with Dynamic Key Negotiation before receiving the network key, and Figure 1-6 Joining in Revision 23 with Dynamic Key Negotiation after receiving the network key.

The Trust Center Link Key passphrase update is only done once the device has received the Network Key and is joined and authorized on the network. This update is a one-time operation. The new passphrase SHALL be kept for the life of the device on the network and SHALL NOT be updated a second time. A device updates the Passphrase for its Trust Center Link Key as follows

1. Send a ZDO Security_Retrieve_Authentication_Token_req to the Trust Center.
    a. Include the Authentication Token ID Local TLV with an ID value of 69, requesting the 128-bit Symmetric Passphrase Global TLV.
2. Follow the processing rules in section 2.4.4.4.2 upon receipt of the ZDO Security_Retrieve_Authentication_rsp command. Notify the application of the change to the passphrase.
3. If no passphrase update occurs within apsSecurityTimeOutPeriod, notify the application.
    a. The application will determine whether to retry this operation or take other action.

4.9.8 Negotiation an Application Link Key with a Partner Device

After joining a network two devices MAY negotiate an application link key using the APSME-KEY-NEGOTIATION primitives. The rules for determining whether partner devices are allowed to negotiate application link keys is outside the scope of this specification. The higher level application MAY impose restrictions on this. When the device receives a request to negotiate an application link key the APSME-KEY-NEGOTIATION.indication is passed to the application. The application can determine if the negotiation is allowed and respond by initiating the APSME-KEY-NEGOTIATION.response.

In a centralized network partner devices can leverage the Trust Center to establish a symmetric passphrase first. This can be used by both sides to authenticate the partner link key. Figure 4-43 shows the exchange between Initiator, Trust Center, and Responder.

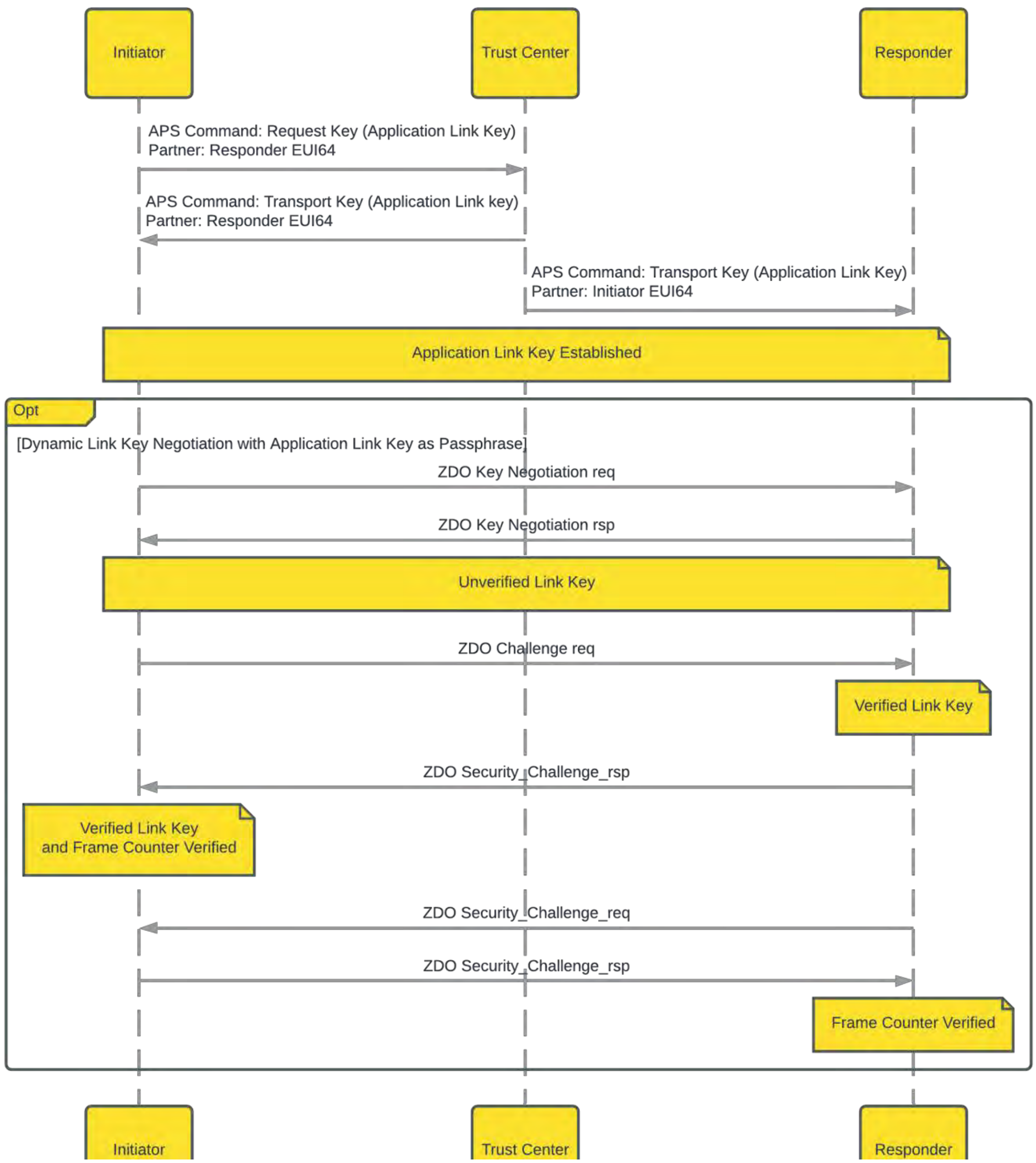


Figure 4-43. Dynamic Application Link Key Establishment

This page intentionally left blank.

ANNEX A CCM* MODE OF OPERATION

CCM* is a generic combined encryption and authentication block cipher mode. CCM* is only defined for use with block ciphers having a 128-bit block size, such as AES-128 [B8]. The CCM* principles can easily be extended to other block sizes but doing so will require further definitions.

The CCM* mode coincides with the original CCM mode specification [B16] for messages that require authentication and, possibly, encryption, but does also offer support for messages that require only encryption. As with the CCM mode, the CCM* mode requires only one key. The security proof for the CCM mode([B17] and [B18]) carries over to the CCM* mode described here. The design of the CCM* mode takes into account the results of [B19], thus allowing it to be securely used in implementation environments in which the use of variable-length authentication tags, rather than fixed-length authentication tags only, is beneficial.

Prerequisites: The following are the prerequisites for the operation of the generic CCM* mode:

1. A block-cipher encryption function *E* SHALL have been chosen, with a 128-bit block size. The length in bits of the keys used by the chosen encryption function is denoted by *keylen*.
2. A fixed representation of octets as binary strings SHALL have been chosen (for example, most-significant-bit first order or least-significant-bit-first order).
3. The length *L* of the message length field, in octets, SHALL have been chosen. Valid values for *L* are the integers 2, 3,..., 8 (the value *L*=1 is reserved).
4. The length *M* of the authentication field, in octets, SHALL have been chosen. Valid values for *M* are the integers 0, 4, 6, 8, 10, 12, 14, and 16. (The value *M*=0 corresponds to disabling authenticity, since then the authentication field contains an empty string.)

A.1 Notation and Representation

Throughout this specification, the representation of integers as octet strings SHALL be fixed. All integers SHALL be represented as octet strings in most-significant-octet first order. This representation conforms to the conventions in Section 4.3 of ANSI X9.63-2001 [B7].

A.2 CCM* Mode Encryption and Authentication Transformation

The CCM* mode forward transformation involves the execution, in order, of an input transformation (A.2.1), an authentication transformation (A.2.2), and encryption transformation (A.2.3).

Input: The CCM* mode forward transformation takes as inputs:

1. A bit string *Key* of length *keylen* bits to be used as the key. Each entity SHALL have evidence that access to this key is restricted to the entity itself and its intended key-sharing group member(s).
2. A nonce *N* of 15-*L* octets. Within the scope of any encryption key *Key*, the nonce value SHALL be unique.
3. An octet string *m* of length *l*(*m*) octets, where $0 \leq l(m) \leq 28L$.
4. An octet string *a* of length *l*(*a*) octets, where $0 \leq l(a) < 2^{64}$.

The nonce *N* SHALL encode the potential values for *M* such that one can uniquely determine from *N* the value of *M* actually used. The exact format of the nonce *N* is outside the scope of this specification and SHALL be determined and fixed by the actual implementation environment of the CCM* mode.

Note: The exact format of the nonce *N* is left to the application, to allow simplified hardware and software implementations in particular settings. Actual implementations of the CCM* mode MAY restrict the values of *M* that are allowed throughout the life-cycle of the encryption key *Key* to a strict subset of those allowed in the generic CCM* mode. If so, the format of the nonce *N* SHALL be such that one can uniquely determine from *N* the actually used value of *M* in that particular subset. In particular, if *M* is fixed and the value *M=0* is not allowed, then there are no restrictions on *N*, in which case the CCM* mode reduces to the CCM mode.

A.2.1 Input Transformation

This step involves the transformation of the input strings *a* and *m* to the strings *AuthData* and *PlainTextData*, to be used by the authentication transformation and the encryption transformation, respectively.

This step involves the following steps, in order:

1. Form the octet string representation *L(a)* of the length *l(a)* of the octet string *a*, as follows:
   a. If *l(a)*=0, then *L(a)* is the empty string.
   b. If $0 < l(a) < 2^{16}-2^{8}$, then *L(a)* is the 2-octets encoding of *l(a)*.
   c. If $2^{16}-2^{8} \leq l(a) < 2^{32}$, then *L(a)* is the right-concatenation of the octet 0xff, the octet 0xfe, and the 4-octets encoding of *l(a)*.
   d. If $2^{32} \leq l(a) < 2^{64}$, then *L(a)* is the right-concatenation of the octet 0xff, the octet 0xff, and the 8-octets encoding of *l(a)*.
2. Right-concatenate the octet string *L(a)* with the octet string *a* itself. Note that the resulting string contains and *a* encoded in a reversible manner.
3. Form the padded message *AddAuthData* by right-concatenating the resulting string with the smallest non-negative number of all-zero octets such that the octet string *AddAuthData* has length divisible by 16.
4. Form the padded message *PlaintextData* by right-concatenating the octet string *m* with the smallest non-negative number of all-zero octets such that the octet string *PlaintextData* has length divisible by 16.
5. Form the message *AuthData* consisting of the octet strings *AddAuthData* and *PlaintextData*:

   AuthData = AddAuthData || PlaintextData

A.2.2 Authentication Transformation

The data *AuthData* that was established above SHALL be tagged using the tagging transformation as follows:

1. Form the 1-octet *Flags* field consisting of the 1-bit *Reserved* field, the 1-bit *Adata* field, and the 3-bit representations of the integers *M* and *L*, as follows:

   Flags = Reserved || Adata || M || L

   Here, the 1-bit *Reserved* field is reserved for future expansions and SHALL be set to '0'. The 1-bit *Adata* field is set to '0' if *l(a)*=0, and set to '1' if *l(a)*>0. The *L* field is the 3-bit representation of the integer *L*-1, in most-significant-bit-first order. The *M* field is the 3-bit representation of the integer (*M*-2)/2 if *M>0* and of the integer 0 if *M*=0, in most-significant-bit-first order.
2. Form the 16-octet B_0 field consisting of the 1-octet *Flags* field defined above, the 15-*L* octet nonce field *N*, and the *L*-octet representation of the length field *l(m)*, as follows:

   $B_0 = Flags \,||\, Nonce\ N \,||\, l(m)$
3. Parse the message *AuthData as* $B_1 \,||\, B_2 \,||\, \ldots \,||B_t$, where each message block B_i is a 16-octet string.

   The CBC-MAC value X_{t+1} is defined by:

   $X_0 := 0_{128};\ X_{i+1} := E(Key, X_i \oplus B_i)$ *for i=0, ... , t.*

   Here, E(*K*, *x*) is the cipher-text that results from encryption of the plaintext *x* using the established block-cipher encryption function *E* with key *Key*; the string 0^{128} is the 16-octet all-zero bit string.

   The authentication tag *T* is the result of omitting all but the leftmost *M* octets of the CBC-MAC value X_{n+1} thus computed.

A.2.3 Encryption Transformation

The data *PlaintextData* that was established in section A.2.1 (step 4) and the authentication tag T that was established in section A.2.2 (step 3) SHALL be encrypted using the encryption transformation as follows:

1. Form the 1-octet *Flags* field consisting of two 1-bit *Reserved* fields, and the 3- bit representations of the integers *0* and L, as follows:

   Flags = Reserved || Reserved || 0 || L

   Here, the two 1-bit *Reserved* fields are reserved for future expansions and SHALL be set to '0'. The L field is the 3-bit representation of the integer L-1, in most-significant- bit-first order. The *'0'* field is the 3-bit representation of the integer 0, in most-significant-bit-first order.

   Define the 16-octet A_i field consisting of the 1-octet *Flags* field defined above, the 15-L octet nonce field N, and the L-octet representation of the integer i, as follows:

   A_i = *Flags || Nonce N || Counter i, for i=0, 1, 2, …*

   Note that this definition ensures that all the A_i fields are distinct from the B_0 fields that are actually used, as those have a *Flags* field with a non-zero encoding of M in the positions where all A_i fields have an all-zero encoding of the integer 0 (see section A.2.2, step 1).

   Parse the message *PlaintextData* as $M_1 \| \ldots \| M_t$, where each message block M_i is a 16-octet string.

   The ciphertext blocks $C_1, \ldots, C_t$ are defined by:

   $C_i := E(Key, A_i) \oplus M_i$ *for i=1, 2, ... , t*

   The string *Ciphertext* is the result of omitting all but the leftmost *l(m)* octets of the string $C_1 \| \ldots \| C_t$

   Define the 16-octet encryption block S_0 by:

   $S_0 := E(Key, A_0)$

2. The encrypted authentication tag U is the result of XOR-ing the string consisting of the leftmost M octets of S_0 and the authentication tag T.

Output: If any of the above operations has failed, then output 'invalid'. Otherwise, output the right-concatenation of the encrypted message *Ciphertext* and the encrypted authentication tag U.

A.3 CCM* Mode Decryption and Authentication Checking Transformation

Input: The CCM* inverse transformation takes as inputs:

1. A bit string *Key* of length *keylen* bits to be used as the key. Each entity SHALL have evidence that access to this key is restricted to the entity itself and its intended key-sharing group member(s).
2. A nonce N of 15-L octets. Within the scope of any encryption key *Key*, the nonce value SHALL be unique.
3. An octet string c of length $l(c)$ octets, where $0 \leq l(c)-M < 2^{8L}$.
4. An octet string a of length $l(a)$ octets, where $0 \leq l(a) < 2^{64}$.

A.3.1 Decryption Transformation

The decryption transformation involves the following steps, in order:

1. Parse the message c as C ||U, where the rightmost string U is an M-octet string. If this operation fails, output 'invalid' and stop. U is the purported encrypted authentication tag. Note that the leftmost string C has length l(c)-M octets.
2. Form the padded message CiphertextData by right-concatenating the string C with the smallest non-negative number of all-zero octets such that the octet string CiphertextData has length divisible by 16.
3. Use the encryption transformation in section A.2.3, with the data CipherTextData and the tag U as inputs.
4. Parse the output string resulting from applying this transformation as m || T, where the rightmost string T is an M-octet string. T is the purported authentication tag. Note that the leftmost string m has length l(c)-M octets.

A.3.2 Authentication Checking Transformation

The authentication checking transformation involves the following steps:

1. Form the message AuthData using the input transformation in section A.2.1, with the string a and the octet string m that was established in section A.3.1 (step 4) as inputs.
2. Use the authentication transformation in section A.2.2, with the message AuthData as input.
3. Compare the output tag MACTag resulting from this transformation with the tag T that was established in section A.3.1 (step 4). If MACTag=T, output ‘valid’; otherwise, output ‘invalid’ and stop.

Output: If any of the above verifications has failed, then output ‘invalid’ and reject the octet string *m*. Otherwise, accept the octet string *m* and accept one of the key sharing group member(s) as the source of *m*.

A.4 Restrictions

All implementations SHALL limit the total amount of data that is encrypted with a single key. The CCM* encryption transformation SHALL invoke not more than 2^{61} block-cipher encryption function operations in total, both for the CBC-MAC and for the CTR encryption operations.

At CCM* decryption, one SHALL verify the (truncated) CBC-MAC before releasing any information, such as, *Plaintext*. If the CBC-MAC verification fails, only the fact that the CBC-MAC verification failed shall be exposed; all other information shall be destroyed.

ANNEX B SECURITY BUILDING BLOCKS

This annex specifies the cryptographic primitives and mechanisms that are used to implement the security protocols in this standard.

B.1 Symmetric-Key Cryptographic Building Blocks

The following symmetric-key cryptographic primitives and data elements are defined for use with all security-processing operations specified in this standard.

B.1.1 Block-Cipher

The block-cipher used in this specification SHALL be the Advanced Encryption Standard AES-128, as specified in FIPS Pub 197. This block-cipher has a key size *keylen* that is equal to the block size, in bits, *i.e.*, *keylen*=128.

B.1.2 Mode of Operation

The block-cipher mode of operation used in this specification SHALL be the CCM* mode of operation, as specified in section A.2.3, with the following instantiations:

1. Each entity SHALL use the block-cipher *E* as specified in section B.1.1.
2. All octets SHALL be represented as specified in the "Conventions and Abbreviations."
3. The parameter *L* SHALL have the integer value 2.
4. The parameter *M* SHALL have one of the following integer values: 0, 4, 8, or 16.

B.1.3 Cryptographic Hash Function

The cryptographic hash function used in this specification SHALL be the blockcipher based cryptographic hash function specified in section B.4, with the following instantiations:

1. Each entity SHALL use the block-cipher *E* as specified in section B.1.1.
2. All integers and octets SHALL be represented as specified in section B.1.2.

The Matyas-Meyer-Oseas hash function (specified in section B.4) has a message digest size *hashlen* that is equal to the block size, in bits, of the established block-cipher.

B.1.4 Keyed Hash Function for Message Authentication

The keyed hash message authentication code (HMAC) used in this specification SHALL be HMAC, as specified in the FIPS Pub 198 [B9], with the following instantiations:

1. Each entity SHALL use the cryptographic hash *H* function as specified in section B.1.3.
2. The block size *B* SHALL have the integer value 16 (this block size specifies the length of the data integrity key, in bytes, that is used by the keyed hash function, *i.e.*, it uses a 128-bit data integrity key).
3. The output size *HMAClen* of the HMAC function SHALL have the same integer value as the message digest parameter *hashlen* as specified in section B.1.3.

B.1.5 Specialized Keyed Hash Function for Message Authentication

The specialized keyed hash message authentication code used in this specification SHALL be as specified in section B.1.4.

B.1.6 Challenge Domain Parameters

The challenge domain parameters used in the specification SHALL be as specified in section B.2.1, with the following instantiation: (*minchallengelen, maxchallengelen*)=(128,128).

All challenges SHALL be validated using the challenge validation primitive as specified in section B.3.

B.2 Challenge Domain Parameter Generation and Validation

This section specifies the primitives that SHALL be used to generate and validate challenge domain parameters.

Challenge domain parameters impose constraints on the length(s) of bit challenges a scheme expects. As such, this establishes a bound on the entropy of challenges and, thereby, on the security of the cryptographic schemes in which these challenges are used. In most schemes, the challenge domain parameters will be such that only challenges of a fixed length will be accepted (for example, 128-bit challenges). However, one MAY define the challenge domain parameters such that challenges of varying length might be accepted. Doing so is useful in contexts in which entities that wish to engage in cryptographic schemes might have a bad random number generator onboard. Allowing both entities that engage in a scheme to contribute sufficiently long inputs enables each of them to contribute sufficient entropy to the scheme.

In this standard, challenge domain parameters will be shared by a number of entities using a scheme determined by the standard. The challenge domain parameters MAY be public; the security of the system does not rely on these parameters being secret.

B.2.1 Challenge Domain Parameter Generation

Challenge domain parameters SHALL be generated using the following routine.

Input: This routine does not take any input.

Actions: The following actions are taken:

1. Choose two nonnegative integers *minchallengelen* and *maxchallengelen*, such that *minchallengelen* ≤ *maxchallengelen*.

Output: Challenge domain parameters *D*=(*minchallengelen*, *maxchallengelen*).

B.2.2 Challenge Domain Parameter Verification

Challenge domain parameters SHALL be verified using the following routine.

Input: Purported set of challenge domain parameters *D*=(*minchallengelen*, *maxchallengelen*).

Actions: The following checks are made:

1. Check that *minchallengelen* and *maxchallengelen* are non-negative integers.
2. Check that *minchallengelen* ≤ *maxchallengelen*.

Output: If any of the above verifications has failed, then output 'invalid' and reject the challenge domain parameters. Otherwise, output 'valid' and accept the challenge domain parameters.

B.3 Challenge Validation Primitive

It is used to check whether a challenge to be used by a scheme in the standard has sufficient length (for example, messages that are too short are discarded, due to insufficient entropy).

Input: The input of the validation transformation is a valid set of challenge domain parameters *D*=(*minchallengelen*, *maxchallengelen*), together with the bit string *Challenge*.

Actions: The following actions are taken:

1. Compute the bit-length *challengelen* of the bit string *Challenge*.
2. Verify that *challengelen* ∈ [*minchallengelen*, *maxchallengelen*]. (That is, verify that the challenge has an appropriate length.)

Output: If the above verification fails, then output 'invalid' and reject the challenge. Otherwise, output 'valid' and accept the challenge.

B.4 Block-Cipher-Based Cryptographic Hash Function

This section specifies the Matyas-Meyer-Oseas hash function, a cryptographic hash function based on block-ciphers. We define this hash function for blockciphers with a key size equal to the block size, such as AES-128, and with a particular choice for the fixed initialization vector *IV* (we take *IV*=0). For a more general definition of the Matyas-Meyer-Oseas hash function, refer to Section 9.4.1 of [B15].

Prerequisites: The following are the prerequisites for the operation of Matyas- Meyer-Oseas hash function:

1. A block-cipher encryption function *E* SHALL have been chosen, with a key size that is equal to the block size. The Matyas-Meyer-Oseas hash function has a message digest size that is equal to the block size of the established encryption function. It operates on bit strings of length less than 22^n, where *n* is the block size, in octets, of the established block-cipher.
2. A fixed representation of integers as binary strings or octet strings SHALL have been chosen.

Input: The input to the Matyas-Meyer-Oseas hash function is as follows:

1. A bit string *M* of length *l* bits, where $0 \le l < 22^n$

Actions: The hash value SHALL be derived as follows:

1. If the message *M* has length less than 2^n bits, pad this message according to the following procedure:
   a. Right-concatenate to the message *M* the binary consisting of the bit '1' followed by *k* '0' bits, where *k* is the smallest non-negative solution to the equation:
   $$l+1+k \equiv 7n\ (\mathbf{mod}\ 8n) \qquad (1)$$
   b. Form the padded message *M'* by right-concatenating to the resulting string the *n*-bit string that is equal to the binary representation of the integer *l*.
2. Otherwise pad this message according to the following method:
   a. Right concatenate to the message *M* the binary consisting of the bit '1' followed by *k* '0' bits, where *k* is the smallest non-negative solution to the equation:
   $$l + 1 + k \equiv 5n\ (mod\ 8n) \qquad (2)$$
   b. Form the padded message *M'* by right-concatenating to the resulting string the 2n-bit string that is equal to the binary representation of the integer l and right-concatenating to the resulting string the n-bit all-zero bit string.
3. Parse the padded message *M'* as $M_1 \,||\, M_2 || \ldots || M_t$ where each message block M_i is an *n*-octet string.
4. The output $Hash_t$ is defined by
   $$Hash_0 = 0^{8n};\ Hash_j = E(Hash_{j-1}, M_j) \oplus M_j \text{ for } j=1,\ldots,t \qquad (3)$$
   Here, $E(K, x)$ is the ciphertext that results from encryption of the plaintext *x*, using the established block-cipher encryption function E with key K; the string 0^{8n} is the *n*-octet all-zero bit string.

Output: The bit string $Hash_t$ as the hash value.

Note that the cryptographic hash function operates on bit strength of length less than 22^n bits, where *n* is the block size (or key size) of the established block cipher, in bytes. For example, the Matyas-Meyer-Oseas hash function with AES- 128 operates on bit strings of length less than 232 bits. It is assumed that all hash function calls are on bit strings of length less than 22^n bits. Any scheme attempting to call the hash function on a bit string exceeding 22^n bits SHALL output 'invalid' and stop.

This page intentionally left blank.

ANNEX C TEST VECTORS FOR CRYPTOGRAPHIC BUILDING BLOCKS

This annex provides sample test vectors for the Zigbee community, aimed at with the intent of assisting in building interoperable security implementations. The sample test vectors are provided as is, pending independent validation.

C.1 Data Conversions

For test vectors, see Appendix J1 of ANSI X9.63-2001 [B7].

C.2 AES Block Cipher

This annex provides sample test vectors for the block-cipher specified in section B.1.1.

For test vectors, see FIPS Pub 197 [B8].

C.3 CCM* Mode Encryption and Authentication Transformation

This annex provides sample test vectors for the mode of operation as specified in section B.1.2.

Prerequisites: The following prerequisites are established for the operation of the mode of operation:

1. The parameter *M* SHALL have the integer value 8.

Input: The inputs to the mode of operation are:

1. The key *Key* of size *keylen*=128 bits to be used:

   Key = C0 C1 C2 C3 C4 C5 C6 C7 C8 C9 CA CB CC CD CE CF

2. The nonce *N* of 15-*L*=13 octets to be used:

   Nonce = A0 A1 A2 A3 A4 A5 A6 A7 || 03 02 01 00 || 06

3. The octet string *m* of length *l(m)=23* octets to be used:

   m = 08 09 0A 0B 0C 0D 0E 0F 10 11 12 13 14 15 16 17 18 19 1A 1B 1C 1D 1E

4. The octet string *a* of length *l(a)*=8 octets to be used:

   a = 00 01 02 03 04 05 06 07

C.3.1 Input Transformation

This step involves the transformation of the input strings *a* and *m* to the strings *AuthData* and *PlainTextData*, to be used by the authentication transformation and the encryption transformation, respectively.

1. Form the octet string representation *L(a)* of the length *l(a)* of the octet string *a*:

   L(a) = 00 08

2. Right-concatenate the octet string *L(a)* and the octet string *a* itself:

   L(a) || a = 00 08 || 00 01 02 03 04 05 06 07

3. Form the padded message AddAuthData by right-concatenating the resulting string with the smallest non-negative number of all-zero octets such that the octet string AddAuthData has length divisible by 16:

   AddAuthData = 00 08 || 00 01 02 03 04 05 06 07 || 00 00 00 00 00 00

4. Form the padded message PlaintextData by right-concatenating the octet string m with the smallest non-negative number of all-zero octets such that the octet string PlaintextData has length divisible by 16:

   PlaintextData = *08 09 0A 0B 0C 0D 0E 0F 10 11 12 13 14 15 16 17 ||*
   18 19 1A 1B 1C 1D 1E || 00 00 00 00 00 00 00 00 00

5. Form the message AuthData consisting of the octet strings AddAuthData and PlaintextData:

   AuthData = 00 08 00 01 02 03 04 05 06 07 00 00 00 00 00 00 ||
   08 09 0A 0B 0C 0D 0E 0F 10 11 12 13 14 15 16 17
   18 19 1A 1B 1C 1D 1E 00 00 00 00 00 00 00 00 00

C.3.2 Authentication Transformation

The data *AuthData* that was established above SHALL be tagged using the following tagging transformation:

1. Form the 1-octet Flags field as follows:

   Flags = 59

2. Form the 16-octet B_0 field as follows:

   B_0 = 59 || A0 A1 A2 A3 A4 A5 A6 A7 03 02 01 00 06 || 00 17

3. Parse the message AuthData as B1 || B2 ||B3, where each message block Bi is a 16-octet string.
4. The CBC-MAC value X4 is calculated as follows:

i	B_i	X_i
0	*59 A0 A1 A2 A3 A4 A5 A6 A7 03 02 01 00 06 00 17*	*00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00*
1	*00 08 00 01 02 03 04 05 06 07 00 00 00 00 00 00*	*F7 74 D1 6E A7 2D C0 B3 E4 5E 36 CA 8F 24 3B 1A*
2	*08 09 0A 0B 0C 0D 0E 0F 10 11 12 13 14 15 16 17*	*90 2E 72 58 AE 5A 4B 5D 85 7A 25 19 F3 C7 3A B3*
3	*18 19 1A 1B 1C 1D 1E 00 00 00 00 00 00 00 00 00*	*5A B2 C8 6E 3E DA 23 D2 7C 49 7D DF 49 BB B4 09*
4	*æ*	*B9 D7 89 67 04 BC FA 20 B2 10 36 74 45 F9 83 D6*

The authentication tag *T* is the result of omitting all but the leftmost *M*=8 octets of the CBC-MAC value X_4:
T = B9 D7 89 67 04 BC FA 20

C.3.3 Encryption Transformation

The data *PlaintextData* SHALL be encrypted using the following encryption transformation:

1. Form the 1-octet Flags field as follows:

   Flags = 01

2. Define the 16-octet Ai field as follows:

i	A_i
0	01 || A0 A1 A2 A3 A4 A5 A6 A7 03 02 01 00 06 || 00 00
1	01 || A0 A1 A2 A3 A4 A5 A6 A7 03 02 01 00 06 || 00 01
2	01 || A0 A1 A2 A3 A4 A5 A6 A7 03 02 01 00 06 || 00 02

3. Parse the message PlaintextData as M1 ||M2, where each message block Mi is a 16-octet string.
4. The ciphertext blocks C1, C2 are computed as follows:

i	AES(Key,A_i)	C_i = AES(Key,A_i) $\oplus$ M_i
1	12 5C A9 61 B7 61 6F 02 16 7A 21 66 70 89 F9 07	1A 55 A3 6A BB 6C 61 0D 06 6B 33 75 64 9C EF 10
2	CC 7F 54 D1 C4 49 B6 35 46 21 46 03 AA C6 2A 17	D4 66 4E CA D8 54 A8 35 46 21 46 03 AA C6 2A 17

5. The string Ciphertext is the result of omitting all but the leftmost l(m)=23 octets of the string C1 ||C2:

   CipherText = 1A 55 A3 6A BB 6C 61 0D 06 6B 33 75 64 9C EF 10 || D4 66 4E CA D8 54 A8

6. Define the 16-octet encryption block S0 by:

   $S_0 = E(Key, A_0)$= *B3 5E D5 A6 DC 43 6E 49 D6 17 2F 54 77 EB B4 39*

7. The encrypted authentication tag U is the result of XOR-ing the string consisting of the leftmost M=8 octets of S0 and the authentication tag T:

   U = 0A 89 5C C1 D8 FF 94 69

   Output: the right-concatenation *c* of the encrypted message *Ciphertext* and the encrypted authentication tag *U*:

   c = 1A 55 A3 6A BB 6C 61 0D 06 6B 33 75 64 9C EF 10 || D4 66 4E CA D8 54 A8 || 0A 89 5C C1 D8 FF 94 69

C.4 CCM* Mode Decryption and Authentication Checking Transformation

This annex provides sample test vectors for the inverse of the mode of operation as specified in section B.1.2.

Prerequisites: The following prerequisites are established for the operation of the mode of operation:

1. The parameter *M* SHALL have the integer value 8.

Input: The inputs to the inverse mode of operation are:

1. The key Key of size keylen=128 bits to be used:

   Key = C0 C1 C2 C3 C4 C5 C6 C7 C8 C9 CA CB CC CD CE CF

2. The nonce N of 15-L=13 octets to be used:

   Nonce = A0 A1 A2 A3 A4 A5 A6 A7 || 03 02 01 00 || 06

3. The octet string c of length l(c)=31 octets to be used:

c = 1A 55 A3 6A BB 6C 61 0D 06 6B 33 75 64 9C EF 10 || D4 66 4E CA D8 54 A8 || 0A 89 5C C1 D8 FF 94 69

4. The octet string a of length l(a)=8 octets to be used:

   a = 00 01 02 03 04 05 06 07

C.4.1 Decryption Transformation

The decryption transformation involves the following steps, in order:

1. Parse the message c as C ||U, where the rightmost string U is an M-octet string:

   C = 1A 55 A3 6A BB 6C 61 0D 06 6B 33 75 64 9C EF 10 || D4 66 4E CA D8 54 A8;
   U = 0A 89 5C C1 D8 FF 94 69

2. Form the padded message CiphertextData by right-concatenating the string C with the smallest non-negative number of all-zero octets such that the octet string CiphertextData has length divisible by 16.

   CipherTextData = 1A 55 A3 6A BB 6C 61 0D 06 6B 33 75 64 9C EF 10 || D4 66 4E CA D8 54 A8 || 00 00 00 00 00 00 00 00

3. Form the 1-octet Flags field as follows:

   Flags = 01

4. Define the 16-octet Ai field as follows:

i	A_i
0	01 || A0 A1 A2 A3 A4 A5 A6 A7 03 02 01 00 06 || 00 00
1	01 || A0 A1 A2 A3 A4 A5 A6 A7 03 02 01 00 06 || 00 01
2	01 || A0 A1 A2 A3 A4 A5 A6 A7 03 02 01 00 06 || 00 02

5. Parse the message *CiphertextData* as $C_1 \| C_2$, where each message block C_i is a 16-octet string.

6. The ciphertext blocks P1, P2 are computed as follows.

I	AES(Key,A_i)	P_i= AES(Key,A_i) ⊕ C_i
1	12 5C A9 61 B7 61 6F 02 16 7A 21 66 70 89 F9 07	08 09 0A 0B 0C 0D 0E 0F 10 11 12 13 14 15 16 17
2	CC 7F 54 D1 C4 49 B6 35 46 21 46 03 AA C6 2A 17	18 19 1A 1B 1C 1D 1E 00 00 00 00 00 00 00 00 00

7. The octet string m is the result of omitting all but the leftmost l(m)=23 octets of the string P1 || P2:

   m = 08 09 0A 0B 0C 0D 0E 0F 10 11 12 13 14 15 16 17 || 18 19 1A 1B 1C 1D 1E

8. Define the 16-octet encryption block S0 by

   $S_0 = E(Key, A_0)$ = B3 5E D5 A6 DC 43 6E 49 D6 17 2F 54 77 EB B4 39

9. The purported authentication tag T is the result of XOR-ing the string consisting of the leftmost M=8 octets of S0 and the octet string U:

   T = B9 D7 89 67 04 BC FA 20

C.4.2 Authentication Checking Transformation

The authentication checking transformation involves the following steps:

1. Form the message AuthData using the input transformation in Input Transformation, with the string a as inputs and the octet string m that was established in section C.4.1 (step7):

   AuthData = 00 08 01 02 03 04 05 06 07 00 00 00 00 00 00 00 ||
   08 09 0A 0B 0C 0D 0E 0F 10 11 12 13 14 15 16 17
   18 19 1A 1B 1C 1D 1E 00 00 00 00 00 00 00 00 00

2. Use the authentication transformation in section C.3.2, with the message AuthData to compute the authentication tag MACTag as input:

   MACTag = B9 D7 89 67 04 BC FA 20

3. Compare the output tag MACTag resulting from this transformation with the tag T that was established in section C.4.1 (step 9):

   T = B9 D7 89 67 04 BC FA 20 = *MACTag*

Output: Since *MACTag*=*T*, output 'valid' and accept the octet string *m* and accept one of the key sharing group member(s) as the source of *m*.

C.5 Cryptographic Hash Function

This section provides sample test vectors for the cryptographic hash function specified in section B.1.3.

C.5.1 Test Vector Set 1

Input: The input to the cryptographic hash function is as follows:

1. The bit string *M* of length *l*=8 bits to be used:

   M=C0

Actions: The hash value SHALL be derived as follows:

1. Pad the message M by right-concatenating to M the bit '1' followed by the smallest non-negative number of '0' bits, such that the resulting string has length 14 (mod 16) octets:

   C0 || 80 00 00 00 00 00 00 00 00 00 00 00 00

2. Form the padded message M' by right-concatenating to the resulting string the 16-bit string that is equal to the binary representation of the integer l:

   M' = C0 || 80 00 00 00 00 00 00 00 00 00 00 00 00 || 00 08

3. Parse the padded message M' as M1, where each message block Mi is a 16-octet string.
4. The hash value Hash1 is computed as follows:

i	Hash$_i$	M$_i$
0	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	æ
1	AE 3A 10 2A 28 D4 3E E0 D4 A0 9E 22 78 8B 20 6C	C0 80 00 00 00 00 00 00 00 00 00 00 00 00 00 08

Output: the 16-octet string *Hash* = $Hash_1$ = AE 3A 10 2A 28 D4 3E E0 D4 A0 9E 22 78 8B 20 6C.

C.5.2 Test Vector Set 2

Input: The input to the cryptographic hash function is as follows:

1. The bit string M of length l=128 bits to be used:

   M=C0 C1 C2 C3 C4 C5 C6 C7 C8 C9 CA CB CC CD CE CF

Actions: The hash value SHALL be derived as follows:

1. Pad the message M by right-concatenating to M the bit ‘1’ followed by the smallest non-negative number of ‘0’ bits, such that the resulting string has length 14 (mod 16) octets:

   C0 C1 C2 C3 C4 C5 C6 C7 C8 C9 CA CB CC CD CE CF ||
   80 00 00 00 00 00 00 00 00 00 00 00 00 00

2. Form the padded message M’ by right-concatenating to the resulting string the 16-bit string that is equal to the binary representation of the integer l:

   M’ = C0 C1 C2 C3 C4 C5 C6 C7 C8 C9 CA CB CC CD CE CF ||
   80 00 00 00 00 00 00 00 00 00 00 00 00 00 || 00 80

3. Parse the padded message M’ as M1 || M2, where each message block Mi is a 16-octet string.
4. The hash value Hash2 is computed as follows:

i	$Hash_i$	M_i
0	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	æ
1	84 EE 75 E5 4F 9A 52 0F 0B 30 9C 35 29 1F 83 4F	C0 C1 C2 C3 C4 C5 C6 C7 C8 C9 CA CB CC CD CE CF
2	A7 97 7E 88 BC 0B 61 E8 21 08 27 10 9A 22 8F 2D	80 00 00 00 00 00 00 00 00 00 00 00 00 00 00 08

Output: the 16-octet string $Hash = Hash_2$ = A7 97 7E 88 BC 0B 61 E8 21 08 27 10 9A 22 8F 2D.

C.5.3 Test Vector Set 3

Input: The input to the cryptographic hash function is as follows:

1. The bit string M of length l = 65528 bits to be used.
2. 8191 bytes (sequence of 0, 1, 2, … 255, 0, 1, 2, …)
3. This test vector is beneath the threshold of a 216 bit string so the first padding method described in section B.4 is utilized.

Actions: The hash value SHALL be derived as follows:

1. Pad the message by right-concatenating to M the bit 1 followed by the smallest non-negative number of '0' bits, such that the resulting string has length 14 (mod 16) octets:

   00 01 02 03 04 … FB FC FD FE || 80 00 00 00 00 00 00 00 00 00 00 00 00 00 00

2. Form the padded message M' by right-concatenating to the resulting string the 16-bit string that is equal to the binary representation of the integer l:

   00 01 02 03 04 … FB FC FD FE || 80 00 00 00 00 00 00 00 00 00 00 00 00 00 00 || FF F8

3. Parse the padded message M' as M1, where each message block Mi is a 16-octet string.

4. The hash value Hash1 is computed as follows using 16-byte hash block operations:

i	$Hash_i$	M_i
0	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	-
1	7A CB 0D DA B8 D3 EA 7B 97 9E 4C 6D 1A EB AC 8D	00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F
...	...	...
i - 1	C3 22 D1 D3 9D 10 86 43 82 06 BD EB 26 41 66 1C	F0 F1 F2 F3 F4 F5 F6 F7 F8 F9 FA FB FC FD FE 80
i	24 EC 2F E7 5B BF FC B3 47 89 BC 06 10 E7 F1 65	00 00 00 00 00 00 00 00 00 00 00 00 00 00 FF F8

C.5.4 Test Vector 4

Input: The input to the cryptographic hash function is as follows:

1. The bit string M of length l = 65536 bits to be used.
2. 8192 bytes (sequence of 0, 1, 2, … 255, 0, 1, 2, …)
3. This test vector is above the threshold of a 216 bit string so the second padding method described in section B.4 is utilized.

Actions: The hash value SHALL be derived as follows.

1. Pad the message by right-concatenating to M the bit 1 followed by the smallest non-negative number of '0' bits, such that the resulting string has length 10 (mod 16) octets:

   00 01 02 03 04 … FB FC FD FE FF || 80 00 00 00 00 00 00 00 00 00

2. Form the padded message M' by right-concatenating to the resulting string the 32-bit string that is equal to the binary representation of the integer l:

   00 01 02 03 04 … FB FC FD FE FF || 80 00 00 00 00 00 00 00 00 00 || 00 01 00 00

3. Concatenate a 16-bit string of zeros for the padding normally used by the first padding method described in section B.4.

   00 01 02 03 04 … FB FC FD FE FF || 80 00 00 00 00 00 00 00 00 00 || 00 01 00 00 || 00 00

4. Parse the padded message M' as M1, where each message block Mi is a 16-octet string.
5. The hash value Hash1 is computed as follows using 16-byte hash block operations:

i	$Hash_i$	M_i
0	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	-
1	7A CB 0D DA B8 D3 EA 7B 97 9E 4C 6D 1A EB AC 8D	00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F
...	...	...
i - 1	4E 55 0D CE 34 31 42 96 41 BA D0 C7 BC 44 34 67	F0 F1 F2 F3 F4 F5 F6 F7 F8 F9 FA FB FC FD FE FF
i	DC 6B 06 87 F0 9F 86 07 13 1C 17 0B 3B D3 15 91	80 00 00 00 00 00 00 00 00 00 00 01 00 00 00 00

C.5.5 Test Vector 5

Input: The input to the cryptographic hash function is as follows:

1. The bit string M of length l = 65608 bits to be used.
2. 8201 bytes (sequence of 0, 1, 2, … 255, 0, 1, 2, …)
3. This test vector is above the threshold of a 216 bit string so the second padding method described in section B.4 is utilized.

Actions: The hash value SHALL be derived as follows.

1. Pad the message by right-concatenating to M the bit 1 followed by the smallest non-negative number of '0' bits, such that the resulting string has length 10 (mod 16) octets:

   00 01 02 03 04 … 04 05 06 07 08 || 80

2. Form the padded message M' by right-concatenating to the resulting string the 32-bit string that is equal to the binary representation of the integer l:

   00 01 02 03 04 … 04 05 06 07 08 || 80 || 00 01 00 48

3. Concatenate a 16-bit string of zeros for the padding normally used by the first padding method described in section B.4.

   00 01 02 03 04 … 04 05 06 07 08 || 80 || 00 01 00 48 || 00 00

4. Parse the padded message M' as M1, where each message block Mi is a 16-octet string.

5. The hash value Hash1 is computed as follows using 16-byte hash block operations:

i	$Hash_i$	M_i
0	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	-
1	7A CB 0D DA B8 D3 EA 7B 97 9E 4C 6D 1A EB AC 8D	00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F
...	...	...
i - 1	4E 55 0D CE 34 31 42 96 41 BA D0 C7 BC 44 34 67	F0 F1 F2 F3 F4 F5 F6 F7 F8 F9 FA FB FC FD FE FF
i	72 C9 B1 5E 17 8A A8 43 E4 A1 6C 58 E3 36 43 A3	00 01 02 03 04 05 06 07 08 80 00 01 00 48 00 00

C.5.6 Test Vector 6

Input: The input to the cryptographic hash function is as follows:

1. The bit string M of length l = 65616 bits to be used.
2. 8202 bytes (sequence of 0, 1, 2, … 255, 0, 1, 2, …)
3. This test vector is above the threshold of a 216 bit string so the second padding method described in section B.4 is utilized.

Actions: The hash value SHALL be derived as follows.

1. Pad the message by right-concatenating to M the bit 1 followed by the smallest non-negative number of '0' bits, such that the resulting string has length 10 (mod 16) octets:

   00 01 02 03 04 … 05 06 07 08 09 || 80 00 00 00 00 00 00 00 00 00 00 00 00 00 00

2. Form the padded message M' by right-concatenating to the resulting string the 32-bit string that is equal to the binary representation of the integer l:

   00 01 02 03 04 … 05 06 07 08 09 || 80 00 00 00 00 00 00 00 00 00 00 00 00 00 00 || 00 01 00 50

3. Concatenate a 16-bit string of zeros for the padding normally used by the first padding method described in section B.4.

   00 01 02 03 04 … 05 06 07 08 09 || 80 00 00 00 00 00 00 00 00 00 00 00 00 00 00 || 00 01 00 50 || 00 00

4. Parse the padded message M' as M1, where each message block Mi is a 16-octet string.

5. The hash value Hash1 is computed as follows using 16-byte hash block operations:

i	$Hash_i$	M_i
0	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	-
1	7A CB 0D DA B8 D3 EA 7B 97 9E 4C 6D 1A EB AC 8D	00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F
...	...	...
i - 1	CC C1 F8 A3 D5 6A 93 20 41 08 10 2B 46 25 0D A7	00 01 02 03 04 05 06 07 08 09 80 00 00 00 00 00
i	BC 98 28 D5 9B 2A A3 23 DA F2 0B E5 F2 E6 65 11	00 00 00 00 00 00 00 00 00 00 00 01 00 50 00 00

C.6 Keyed Hash Function for Message Authentication

This annex provides sample test vectors for the keyed hash function for message authentication as specified in section B.1.4.

C.6.1 Test Vector Set 1

Input: The input to the keyed hash function is as follows:

1. The key Key of size keylen=128 bits to be used:

   Key = 40 41 42 43 44 45 46 47 48 49 4A 4B 4C 4D 4E 4F

2. The bit string M of length l=8 bits to be used:

   M=C0

Actions: The keyed hash value SHALL be derived as follows:

1. Create the 16-octet string ipad (inner pad) as follows:

   ipad = 36 36 36 36 36 36 36 36 36 36 36 36 36 36 36 36

2. Form the inner key Key1 by XOR-ing the bit string Key and the octet string ipad:

   $Key_1 = Key \oplus ipad$ = 76 77 74 75 72 73 70 71 7E 7F 7C 7D 7A 7B 78 79

3. Form the padded message M1 by right-concatenating the bit string Key1 with the bit string M:

   $M_1 = Key_1 \| M$ = 76 77 74 75 72 73 70 71 7E 7F 7C 7D 7A 7B 78 79 || C0

4. Compute the hash value Hash1 of the bit string M1:

   $Hash_1$ = 3C 3D 53 75 29 A7 A9 A0 3F 66 9D CD 88 6C B5 2C

5. Create the 16-octet string opad (outer pad) as follows:

   opad = 5C 5C 5C 5C 5C 5C 5C 5C 5C 5C 5C 5C 5C 5C 5C 5C

6. Form the outer key Key2 by XOR-ing the bit string Key and the octet string opad:

   $Key_2 = Key \oplus opad$ = 1C 1D 1E 1F 18 19 1A 1B 14 15 16 17 10 11 12 13

7. Form the padded message M2 by right-concatenating the bit string Key2 with the bit string Hash1:

   $M_2 = Key_2 \| Hash_1$ = 1C 1D 1E 1F 18 19 1A 1B 14 15 16 17 10 11 12 13 ||
   3C 3D 53 75 29 A7 A9 A0 3F 66 9D CD 88 6C B5 2C

8. Compute the hash value Hash2 of the bit string M2:

   $Hash_2$ = 45 12 80 7B F9 4C B3 40 0F 0E 2C 25 FB 76 E9 99

Output: the 16-octet string $HMAC = Hash_2$ = 45 12 80 7B F9 4C B3 40 0F 0E 2C 25 FB 76 E9 99

C.6.2 Test Vector Set 2

Input: The input to the keyed hash function is as follows:

1. The key Key of size keylen=256 bits to be used:

   Key = 40 41 42 43 44 45 46 47 48 49 4A 4B 4C 4D 4E 4F || 50 51 52 53 54 55 56 57 58 59 5A 5B 5C 5D 5E 5F

2. The bit string M of length l=128 bits to be used:

   M = C0 C1 C2 C3 C4 C5 C6 C7 C8 C9 CA CB CC CD CE CF

Actions: The keyed hash value SHALL be derived as follows:

1. Compute the hash value Key0 of the bit string Key:

   Key_0 = 22 F4 0C BE 15 66 AC CF EB 77 77 E1 C4 A9 BB 43

2. Create the 16-octet string ipad (inner pad) as follows:

   $ipad$ = 36 36 36 36 36 36 36 36 36 36 36 36 36 36 36 36

3. Form the inner key Key1 by XOR-ing the bit key Key0 and the octet string ipad:

   $Key_1 = Key_0 \oplus ipad$ = 14 C2 3A 88 23 50 9A F9 DD 41 41 D7 F2 9F 8D 75

4. Form the padded message M1 by right-concatenating the bit string Key1 with the bit string M:

   $M_1 = Key_1 \| M$ = 14 C2 3A 88 23 50 9A F9 DD 41 41 D7 F2 9F 8D 75 ||
   C0 C1 C2 C3 C4 C5 C6 C7 C8 C9 CA CB CC CD CE CF

5. Compute the hash value Hash1 of the bit string M1:

   $Hash_1$ = 42 65 BE 29 74 55 8C A2 7B 77 85 AC 73 F2 22 10

6. Create the 16-octet string opad (outer pad) as follows:

   $opad$ = 5C 5C 5C 5C 5C 5C 5C 5C 5C 5C 5C 5C 5C 5C 5C 5C

7. Form the outer key Key2 by XOR-ing the bit string Key0 and the octet string opad:

   $Key_2 = Key_0 \oplus opad$ = 7E A8 50 E2 49 3A F0 93 B7 2B 2B BD 98 F5 E7 1F

8. Form the padded message M2 by right-concatenating the bit string Key2 with the bit string Hash1:

   $M_2 = Key_2 \| Hash_1$ = 7E A8 50 E2 49 3A F0 93 B7 2B 2B BD 98 F5 E7 1F ||
   42 65 BE 29 74 55 8C A2 7B 77 85 AC 73 F2 22 10

9. Compute the hash value Hash2 of the bit string M2:

   $Hash_2$ = A3 B0 07 99 84 BF 15 57 F7 4A 0D 63 87 E0 A1 1A

Output: the 16-octet string $HMAC = Hash_2$ = A3 B0 07 99 84 BF 15 57 F7 4A 0D 63 87 E0 A1 1A

C.6.3 Specialized Keyed Hash Function for Message Authentication

This annex provides sample test vectors for the specialized keyed hash function for message authentication as specified in section B.1.4.

For test vectors, see section C.6.

C.7 Key Agreement using Elliptic Curve Diffie Helmann Derivatives

Important Note: In this section, binary octet streams are represented with the first octet (for example, 41 hex in the first Private Key below) in the first location in memory and subsequent octets in successive locations in memory.

C.7.1 Test Vectors for SPEKE/Curve25519/AES-MMO-128/HMAC-AES-MMO-128

This section provides test vectors for the Simple Password Exponential Key Exchange (SPEKE) Password-Authenticated Key Exchange (PAKE) algorithm as specified in section J.1.3. In order to generate reproducible results the private keys of the two parties engaging in key agreement, henceforth referred to as Alice and Bob, are not generated randomly from a cryptographically safe source, as they usually would be, but they are rather set as follows:

Party	Chosen Private Key (ASCII)	Private Key (Binary)
Alice	`AliceAliceAliceAliceAliceAliceAl`	`41 6C 69 63 65 41 6C 69 63 65 41 6C 69 63 65 41 6C 69 63 65 41 6C 69 63 65 41 6C 69 63 65 41 6C`
Bob	`BobBobBobBobBobBobBobBobBobBobBo`	`42 6F 62 42 6F 62 42 6F 62 42 6F 62 42 6F 62 42 6F 62 42 6F 62 42 6F 62 42 6F 62 42 6F 62 42 6F`

These keys undergo Curve25519 private key clamping as follows:

Party	Chosen Private Key (ASCII)	Clamped Private Key d (Binary)
Alice (d_i)	`AliceAliceAliceAliceAliceAliceAl`	`40 6C 69 63 65 41 6C 69 63 65 41 6C 69 63 65 41 6C 69 63 65 41 6C 69 63 65 41 6C 69 63 65 41 6C`
Bob (d_r)	`BobBobBobBobBobBobBobBobBobBobBo`	`40 6F 62 42 6F 62 42 6F 62 42 6F 62 42 6F 62 42 6F 62 42 6F 62 42 6F 62 42 6F 62 42 6F 62 42 6F`

Notice that the binary octet stream printed above corresponds to a little-endian representation, i.e., if the key were considered a large integer, its value would be 6C416563…696C40 for Alice, for example.

Party	Identification (EUI-64) A
Alice (A_i)	`A0:A0:0A:0A:A0:A0:00:AA`
Bob (A_r)	`B0:B0:0B:0B:B0:B0:00:BB`

C.7.1.1 Test Vector #1

For the present test vector, the Pre-shared Key (PSK) is assumed as *apscWellKnownPSK*, the well-known key for anonymous key exchanges when there is no installation code derived pre-configured link-key available. This results in the following generator base-point G:

```
G = H*(PSK) = 90 2B 44 85 C8 4E C4 A0 59 44 AB 34 42 92 68 78 ||
              90 2B 44 85 C8 4E C4 A0 59 44 AB 34 42 92 68 78
```

The public points that result for the given private keys on Curve25519 are calculated as Q = dG, where G is the generator base-point as noted above:

Party	Public Key Point Q
Alice (Q_i)	BA BB C4 D7 85 6A BF 56 1B B4 37 8F D9 FD 24 92 C1 EA 16 02 1B 90 D6 1F CE 3A 96 5B 04 1C A2 59
Bob (Q_r)	47 7D C0 5F F5 42 AC 83 AD DF 2B 87 87 11 92 DC 6C 59 6A C5 40 C8 D3 5A FB 7E C7 25 9A 71 5B 6C

Alice and Bob calculate the shared SPEKE (ECDH) secret point via point-multiplication on the curve, by multiplying the remote public point times the local private key, both arriving at the same value:

```
xk = FB 9C 3C 7A 2E 49 03 CD D2 36 DA 82 CD 0B 71 81 B1 61 7D 99 67 4C 4E A8 A3 F5 D4 60 31 DD A7 09
```

For calculating the session identifier I the identifications and public points of initiator (Alice) and responder (Bob) are concatenated; the concatenation order depends on the integer comparison of both identifications (EUI-64s)::

```
Ai || Qi = AA 00 A0 A0 0A 0A A0 A0 ||
BA BB C4 D7 85 6A BF 56 1B B4 37 8F D9 FD 24 92 C1 EA 16 02 1B 90 D6 1F CE 3A 96 5B 04 1C A2 59

Ar || Qr = BB 00 B0 B0 0B 0B B0 B0 ||
47 7D C0 5F F5 42 AC 83 AD DF 2B 87 87 11 92 DC 6C 59 6A C5 40 C8 D3 5A FB 7E C7 25 9A 71 5B 6C

I = AA 00 A0 A0 0A 0A A0 A0
BA BB C4 D7 85 6A BF 56 1B B4 37 8F D9 FD 24 92 C1 EA 16 02 1B 90 D6 1F CE 3A 96 5B 04 1C A2 59
BB 00 B0 B0 0B 0B B0 B0
47 7D C0 5F F5 42 AC 83 AD DF 2B 87 87 11 92 DC 6C 59 6A C5 40 C8 D3 5A FB 7E C7 25 9A 71 5B 6C
```

The final SPEKE shared secret is obtained by concatenating above x-coordinate, the session identifier and the based point G, which was derived from the pre-shared secret (PSK) and calculating the SHA-256 hash value:

```
s = H(xk || I || G) = AE 46 6B 75 30 9D 5C 2D 71 5F 7E 44 31 DF 04 7A
```

The derived APS link-key is then finally:

```
KDF(s, { 0x01 }) = HMAC-AES-MMO-128(s, { 0x01 }) =
```

`25 47 F3 AF 96 39 1E 1E BF F2 A3 B7 6D 6A 29 29`

C.7.1.2 Test Vector #2

For the present test vector, the Pre-shared Key (PSK) is assumed as “ZigBeeAlliance20”. This results in the following generator base-point G:

```
G = H*(PSK) = DE E6 39 E5 FF F9 46 D7 B1 00 CC 5F 3F 9C E8 9C ||
DE E6 39 E5 FF F9 46 D7 B1 00 CC 5F 3F 9C E8 9C

G[0] = 09

G = 09 D0 7B 39 6D 10 C1 48 E3 DA C6 DE A7 DC B2 82
BB DD EE 80 63 63 A6 5C 2B 9F C9 72 7F E2 D4 F0
```

Notice that this base point is larger than the prime $p = 2^{255} - 19$. This test vector has specifically been chosen to ensure that implementations do not fail, when they encounter such basepoints, and perform as expected.

The public points that result for the given private keys on Curve25519 are calculated as Q = dG, where G is the generator base-point as noted above:

Party	Public Key Point Q
Alice (Q_i)	04 8E 8D 32 31 21 96 39 28 21 7B 2C F3 C7 DB 23 AA 4E 75 66 D9 69 BF 0E D5 FC A9 F1 A2 3E 0F 6E
Bob (Q_r)	02 29 D3 63 E4 C5 E6 7F C4 B1 0B E2 CD 98 E4 53 E8 6D 86 33 8E 15 A0 3A 3A 68 A4 83 3F E7 84 0C

Alice and Bob calculate the shared SPEKE (ECDH) secret point via point-multiplication on the curve, by multiplying the remote public point times the local private key, both arriving at the same value:

```
xk = 78 3F 96 76 C9 C7 4A 69 C1 41 C3 C2 7B B9 B4 64 55 12 E7 1B C6 E1 76 79 B4 BC 33 E7 48 5B F5 04
```

For calculating the session identifier I the identifications and public points of initiator (Alice) and responder (Bob) are concatenated; the concatenation order depends on the integer comparison of both identifications (EUI-64s)::

```
Ai || Qi = AA 00 A0 A0 0A 0A A0 A0 ||
04 8E 8D 32 31 21 96 39 28 21 7B 2C F3 C7 DB 23 AA 4E 75 66 D9 69 BF 0E D5 FC A9 F1 A2 3E 0F 6E


Ar || Qr = BB 00 B0 B0 0B 0B B0 B0 ||
02 29 D3 63 E4 C5 E6 7F C4 B1 0B E2 CD 98 E4 53 E8 6D 86 33 8E 15 A0 3A 3A 68 A4 83 3F E7 84 0C

I = AA 00 A0 A0 0A 0A A0 A0
04 8E 8D 32 31 21 96 39 28 21 7B 2C F3 C7 DB 23 AA 4E 75 66 D9 69 BF 0E D5 FC A9 F1 A2 3E 0F 6E
BB 00 B0 B0 0B 0B B0 B0
02 29 D3 63 E4 C5 E6 7F C4 B1 0B E2 CD 98 E4 53 E8 6D 86 33 8E 15 A0 3A 3A 68 A4 83 3F E7 84 0C
```

The final SPEKE shared secret is obtained by concatenating above x-coordinate, the session identifier and the based point G, which was derived from the pre-shared secret (PSK) and calculating the AES-MMO-128 hash value:

```
s = H(xk || I || G) = D9 7B 98 45 C6 06 E1 05 72 8E 84 C1 48 6A 9F 68
```

The derived APS link-key is then finally:

```
KDF(s, { 0x01 }) = HMAC-AES-MMO-128(s, { 0x01 }) =
63 F2 38 9B 06 38 72 20 96 48 18 65 7B 47 B5 DB
```

C.7.2 Test Vectors for Curve25519/SHA-256/HMAC-SHA-256

This section provides test vectors for the password-authenticated key exchange (PAKE) algorithm as specified in section J.1.3. In order to generate reproducible results, the private keys of the two parties engaging in key agreement, henceforth referred to as Alice and Bob, are not generated randomly from a cryptographically safe source, as they usually would be, but they are rather set as follows:

Party	Chosen Private Key (ASCII)	Private Key (Binary)
Alice	AliceAliceAliceAliceAliceAliceAl	41 6C 69 63 65 41 6C 69 63 65 41 6C 69 63 65 41 6C 69 63 65 41 6C 69 63 65 41 6C 69 63 65 41 6C
Bob	BobBobBobBobBobBobBobBobBobBobBo	42 6F 62 42 6F 62 42 6F 62 42 6F 62 42 6F 62 42 6F 62 42 6F 62 42 6F 62 42 6F 62 42 6F 62 42 6F

These keys undergo Curve25519 private key clamping as follows:

Party	Chosen Private Key (ASCII)	Clamped Private Key d (Binary)
Alice (d_i)	`AliceAliceAliceAliceAliceAliceAl`	`40 6C 69 63 65 41 6C 69 63 65 41 6C 69 63 65 41 6C 69 63 65 41 6C 69 63 65 41 6C 69 63 65 41 6C`
Bob (d_r)	`BobBobBobBobBobBobBobBobBobBobBo`	`40 6F 62 42 6F 62 42 6F 62 42 6F 62 42 6F 62 42 6F 62 42 6F 62 42 6F 62 42 6F 62 42 6F 62 42 6F`

Notice that the binary octet stream printed above corresponds to a little-endian representation, i.e. if the key were considered a large integer, its value would be 6C416563…696C40 for Alice, for example.

Party	Identification (EUI-64) A
Alice (A_i)	`A0:A0:0A:0A:A0:A0:00:AA`
Bob (A_r)	`B0:B0:0B:0B:B0:B0:00:BB`

C.7.2.1 Test Vector #1

For the present test vector, the Pre-shared Key (PSK) is assumed as *apscWellKnownPSK*, the well-known key for anonymous key exchanges when there is no installation code derived pre-configured link-key available. This results in the following generator base-point G:

```
G = H(PSK) = EE E8 B7 90 39 6F 5B C0 99 4B E4 4F A7 3C AE 1A
             4B FE DC 7A 71 2F 7B 32 86 4B 45 BF 86 F9 F1 78
```

To avoid collision with a number of known, weak generator points the first byte is set to 0x09. This results in:

```
G[0] = 09
G = 09 E8 B7 90 39 6F 5B C0 99 4B E4 4F A7 3C AE 1A
    4B FE DC 7A 71 2F 7B 32 86 4B 45 BF 86 F9 F1 78
```

The public points that result for the given private keys on Curve25519 are calculated as Q = dG, where G is the generator base-point as noted above:

Party	Public Key Point Q
Alice (Q_i)	`02 B0 9E DD 3B 8D E0 88 4F C4 E1 93 A2 AE 17 97` `02 F8 73 6C 79 F0 E2 34 2C 09 9C 4B F5 F9 B2 77`
Bob (Q_r)	`68 8F A2 A3 F0 D1 92 B4 64 6F E6 62 99 4D 27 CC` `5B 6A 57 71 BA 56 F3 3F 3E 8B ED 5E 71 71 09 0F`

Alice and Bob calculate the shared ECDH secret point via point-multiplication on the curve, by multiplying the remote public point times the local private key, both arriving at the same value:

```
xk = 24 14 1C 4C 06 A2 E7 D5 9F 15 36 7F AC E3 9E C2 0C 17 67 BB 97 25 79 41 6F 14 10 CC 36 22 2A 2F
```

For calculating the session identifier I the identifications and public points of initiator (Alice) and responder (Bob) are concatenated; the concatenation order depends on the integer comparison of both identifications (EUI-64s):


```
Ai || Qi = AA 00 A0 A0 0A 0A A0 A0 ||
02 B0 9E DD 3B 8D E0 88 4F C4 E1 93 A2 AE 17 97 02 F8 73 6C 79 F0 E2 34 2C 09 9C 4B F5 F9 B2 77

Ar || Qr = BB 00 B0 B0 0B 0B B0 B0 ||
68 8F A2 A3 F0 D1 92 B4 64 6F E6 62 99 4D 27 CC 5B 6A 57 71 BA 56 F3 3F 3E 8B ED 5E 71 71 09 0F

I = AA 00 A0 A0 0A 0A A0 A0
02 B0 9E DD 3B 8D E0 88 4F C4 E1 93 A2 AE 17 97 02 F8 73 6C 79 F0 E2 34 2C 09 9C 4B F5 F9 B2 77
BB 00 B0 B0 0B 0B B0 B0
68 8F A2 A3 F0 D1 92 B4 64 6F E6 62 99 4D 27 CC 5B 6A 57 71 BA 56 F3 3F 3E 8B ED 5E 71 71 09 0F
```

The final shared secret is obtained by concatenating above x-coordinate, the session identifier and the based point G, which was derived from the pre-shared secret (PSK) and calculating the SHA-256 hash value:

```
s = H(xk || I || G) = 6F 79 F3 60 C9 8A E7 F2 4E 6D DD AB B5 A3 6D 6E
A8 0C 76 5A C6 91 1B AE 17 1F 21 20 3E 88 90 AD
```

The derived APS link-key is then finally:

```
KDF(s, { 0x01 }) = HMAC-SHA-256(s, { 0x01 }) =
EB 52 E5 BF 6B 5A C7 F0 A9 44 0C AD 78 0B B7 0B
```

Note: The HMAC is truncated to the first 128-bits.

C.7.2.2 Test Vector #2

For the present test vector, the Pre-shared Key (PSK) is assumed as “ZigBeeAlliance20”. This results in the following generator base-point G:

```
G = H(PSK) = F0 D0 7B 39 6D 10 C1 48 E3 DA C6 DE A7 DC B2 82
BB DD EE 80 63 63 A6 5C 2B 9F C9 72 7F E2 D4 F0
```

To avoid collision with a number of known, weak generator points the first byte is set to 0x09. This results in:

```
G[0] = 09
G = 09 D0 7B 39 6D 10 C1 48 E3 DA C6 DE A7 DC B2 82
BB DD EE 80 63 63 A6 5C 2B 9F C9 72 7F E2 D4 F0
```

Notice that this base point is larger than the prime $p = 2^{255} - 19$. This test vector has specifically been chosen to ensure that implementations do not fail, when they encounter such basepoints, and perform as expected.

The public points that result for the given private keys on Curve25519 are calculated as Q = dG, where G is the generator base-point as noted above:

Party	Public Key Point Q
Alice (Q_i)	CF 65 21 5E 9A 4A C0 15 AD 5B 1E 08 70 54 24 DA 83 94 6C 7B 80 7A B1 9F FD D0 3 C 2F 6F B6 37 58
Bob (Q_r)	AA F7 27 B1 F7 9D 63 4C 21 DA 31 E4 AF 39 FD 52 62 92 37 BF 53 C8 B2 03 A5 6C 4B 17 BB 3F B5 51

Alice and Bob calculate the shared secret point via point-multiplication on the curve, by multiplying the remote public point times the local private key, both arriving at the same value:

```
xk = 35 88 71 B8 F6 22 24 4D 9A CF 82 62 47 13 7F 88 9F AF 39 38 A5 38 DC 41 7D E2 E4 14 BB E6 0C 2D
```


For calculating the session identifier I the identifications and public points of initiator (Alice) and responder (Bob) are concatenated; the concatenation order depends on the integer comparison of both identifications (EUI-64s)::

```
Ai || Qi = AA 00 A0 A0 0A 0A A0 A0 ||

CF 65 21 5E 9A 4A C0 15 AD 5B 1E 08 70 54 24 DA 83 94 6C 7B 80 7A B1 9F FD D0 3C 2F 6F B6 37 58


Ar || Qr = BB 00 B0 B0 0B 0B B0 B0 ||
AA F7 27 B1 F7 9D 63 4C 21 DA 31 E4 AF 39 FD 52 62 92 37 BF 53 C8 B2 03 A5 6C 4B 17 BB 3F B5 51


I = AA 00 A0 A0 0A 0A A0 A0
09 D0 7B 39 6D 10 C1 48 E3 DA C6 DE A7 DC B2 82 BB DD EE 80 63 63 A6 5C 2B 9F C9 72 7F E2 D4 F0
BB 00 B0 B0 0B 0B B0 B0
AA F7 27 B1 F7 9D 63 4C 21 DA 31 E4 AF 39 FD 52 62 92 37 BF 53 C8 B2 03 A5 6C 4B 17 BB 3F B5 51
```

The final shared secret is obtained by concatenating above x-coordinate, the session identifier and the based point G, which was derived from the pre-shared secret (PSK) and calculating the SHA-256 hash value:

```
s = H(xk || I || G) = B7 7D 9E A0 B6 4D 6A D7 33 9A B2 7D 60 03 E6 8B
8A 27 26 19 7F 76 E2 0F B8 24 C4 3B 6B B4 5A 47
```

The derived APS link-key is then finally:

```
KDF(s, { 0x01 }) = HMAC-SHA-256(s, { 0x01 }) =
31 ED 35 83 AE 18 89 BE B4 24 90 1B CB 18 1A 99
```

ANNEX D MAC AND PHY SUB-LAYER CLARIFICATIONS

D.1 Introduction

D.1.1 Scope

This annex applies to the IEEE Std 802.15.4-2020 Medium Access Control sub-layer (MAC) and Physical Layer (PHY) specification when used in conjunction with higher layers defined by the Zigbee specification. Nothing is implied about the usage under other circumstances.

D.1.2 Purpose

The current Zigbee specification assumes the use of the MAC and PHY sub-layers defined in the IEEE Std 802.15.4-2020 specification. However, as developers have put the MAC and PHY sub-layers into use, they have uncovered problems that MAY or MAY NOT have been anticipated by the authors of the specification, or are not covered in the IEEE Std 802.15.4-2020 specification. This document is intended to provide solutions to such problems, for use by the Connectivity Standards Alliance.

D.2 Numeric Status Code Values

Some Zigbee over-the-air messages contain MAC layer status code values. IEEE Std 802.15.4-2003 and -2006 contained numeric values for symbolic MAC/PHY status code enumerations, whereas later revisions did not include such values any more. MAC layer status codes, when sent over the air, SHALL use the numeric values provided in IEEE Std 802.15.4-2006, as far as such numeric values exist.

D.3 Stack Size Issues

Both MAC and Zigbee stack developers have discovered that implementation of a full-blown MAC is a major undertaking and requires a great deal of code space. Even with the optional GTS and MAC security features eliminated, it is not surprising to find the MAC taking up more than 24K of code space on a processor with 64K of available space.

The Connectivity Standards Alliance has adopted a compensating policy to declare MAC features that are not required to support a particular stack profile optional with respect to that stack profile. In particular, any MAC feature that will not be exploited as a result of platform compliance testing for a particular stack profile need not be present in order for an implementation to be declared platform compliant. For example, since the Zigbee Pro stack profile relies on a beaconless network, the platform compliance testing for the stack profile does not employ beaconing. The code to support regular beaconing, beacon track, and so on, MAY therefore be absent from the code base of the device under test without the knowledge of the testers, without presenting a problem with respect to platform compliance certification.

The exact list of MAC features that SHALL be supported in a platform is described in the PICS document used for MAC conformance testing.

D.4 MAC Association

At association time, according to the IEEE Std 802.15.4 specification, a number of frames are sent, including an association request command, an associate response command and a data request. There is some ambiguity in the specification regarding the addressing fields in the headers for these frames. Table D-1 to Table D-3 outline the allowable options that SHALL be recognized by devices implementing the Zigbee specification. In each case, the first option given is the preferred option and SHOULD be used.

13689 **Table D-1. Associate Request Header FieldsHeader Fields**

DstPANId	DstAddr	SrcPANId	SrcAddr
The PANId of the destination device.	The 16-bit short address of the destination device.	0xffff	The 64-bit extended address of the source device.
		PANId omitted because the IntraPAN sub-field in the frame control field is set to one.	
		The PANId of the destination device.	
Not present if the destination device is the PAN coordinator.	Not present if the destination device is the PAN coordinator.		

13690 Note that in this case and the case below, the source of the command is the device requesting association.

13691 **Table D-2. Data Request Header Fields**

DstPANId	DstAddr	SrcPANId	SrcAddr
The PANId of the destination device.	The 16-bit short address of the destination device.	0xffff	The 64-bit extended address of the source device.
		PANId omitted because the IntraPAN sub-field in the frame control field is set to one.	
		The PANId of the destination device.	
Not present if the destination device is the PAN coordinator.	Not present if the destination device is the PAN coordinator.		

13692 **Table D-3. Association Response Header Fields**

DstPANId	DstAddr	SrcPANId	SrcAddr
The PANId of the destination device.	The 64-bit extended address of the destination device.	PANId omitted because the IntraPAN sub-field in the frame control field is set to one.	The 64-bit extended address of the source device.

DstPANId	DstAddr	SrcPANId	SrcAddr
		The PANId of the source device.	
0xffff			

D.5 aMaxMACFrameSize

The IEEE Std 802.15.4-2020 MAC specification [B1] has two constants that define the minimum and maximum values for the MAC data packet payload size. These are the *aMaxMACPayloadSize* (118 bytes) and the *aMaxMACSafePayloadSize* (102 bytes). Since the overhead imposed by the MAC header is variable, the actual limit of the MAC data payload size is in between these values and MAY vary by implementation.

When used in a Zigbee platform, the MAC implementation SHALL support transmission and reception of unsecured MAC data packet payloads of up to (*aMaxPHYPacketSize - nwkcMinHeaderOverhead*) bytes. The value of *nwkcMinHeaderOverhead* parameter takes into account the fact that Zigbee uses short addressing modes and intra-PAN communications.

D.6 Frame Version Value

The MAC specification requires that any unsecured MAC data packet with payload size greater than aMaxMACSafePayloadSize (102bytes) SHALL have the Frame Version field set to one (see section 6.3.1 of [B1]). When used in a Zigbee platform, the MAC implementation SHALL always set the Frame Version field in unsecured MAC data packets to zero. The reason for this is to ensure backwards compatibility with existing deployed Zigbee devices that cannot receive packets correctly if these bits are set to a non-zero value. Note that this deviation is only on the transmit side, the receive side processing is unchanged. That is, the MAC implementation SHALL be able to receive and process MAC data packets with the Frame Version field set to any non-reserved value, as specified in section 5.6.1.2 of [B1].

The MAC specification allows the coordinator realignment command to be sent with either Frame Version of zero or one. The format of the command is different in each case (see section 5.3.8.1 of [B1]). When used in a Zigbee implementation, the MAC implementation SHALL always set the Frame Version field in the coordinator realignment command to zero.

D.7 Beaconing in Zigbee Networks

Zigbee SHALL not use periodic beaconing. Beacons are sent in response to beacon requests. Zigbee does not make use of GTS. The following MAC PIB values SHALL be set by the Zigbee stack.

PIB	Value
macBeaconOrder	SHALL be set to 0x0F
macSuperframeOrder	SHALL be set to 0x0F

D.8 CSMA Backoff Timing

The IEEE Std 802.15.4-2020 specification provides an increase in *macMaxBE* to 8 from 5. This higher value is allowed within Zigbee and it is recommended as the default. The default value of *macMinBE* SHOULD be 5 instead of

3. This provides better joining performance in dense networks where many devices MAY be responding to a beacon request. For NA Regional Sub-GHz FSK PHY it is recommended to increase macMinBE to 7 and macMaxBE to 10.

D.9 Recommended Scan Duration

The time a device listens for beacons is set by IEEE Std 802.15.4 to *aBaseSuperframeDuration**(2^n+1) symbols where n is the value of the *ScanDuration* parameter. For Zigbee implementations the value of n SHOULD be set to ensure the duration of the listening window is similar to the length of time the beacon responses are EXPECTED.

For the 2.4GHz a *ScanDuration* value of 3 is recommended. For GB SE and Sub-GHz FSK PHY's and Regional Sub-GHz FSK PHY a *ScanDuration* value of 5 is required.

D.10 MAC Interface Changes

The IEEE Std 802-15-4-2020 specification has no notification when a MAC data poll is received by a coordinator (FFD) or any ability for the Zigbee layers to dictate the response to the MAC data poll. Therefore the following interfaces are defined for a MAC used by Zigbee network layers.

D.10.1 Additional Primitives accessed through the MLME-SAP

Those primitives marked with a diamond (◊) are optional for an RFD.

Name	Request	Indication	Response	Confirm
MLME-Poll	(Already specified in reference [B1])	D.10.2	-	(Already specified in reference [B1])

D.10.2 MLME-POLL.indication

The MLME-Poll.indication primitive notifies the next higher level that a request for data has been received.

D.10.2.1 Semantics of the Service Primitive

The semantics of the MLME-Poll.indication primitive is as follows.

```
MLME-Poll.indication        {
                            AddrMode
                            DeviceAddress
                            }
```

Name	Type	Valid Range	Description
AddrMode	Integer	0x02 – 0x03	This value can take one of the following values: 2=16 bit short address. 3=64 bit extended address.
DeviceAddress	Integer	As specified by AddrMode parameter.	The address of the device requesting pending data.

D.10.2.2 When Generated

The MLME-POLL.indication primitive indicates the reception of a Data request command frame by the MAC sublayer and issued to the local SSCS (service specific convergence sublayer).

D.10.2.3 Effect on Receipt

The effect on receipt of the MLME-Poll.indication primitive is that the next higher layer is notified that a device is requesting to see if there is a pending MAC data frame. If an indirect frame is queued by the higher layer during the processing of an MLME-POLL.indication it SHALL affect the pending bit in the ACK frame corresponding to the data request frame that caused the MLME-POLL.indication to be issued.

D.11 MAC Feature Enhancements

This section describes the MAC Enhancements Applicable to Sub-GHz Only mode, optional for 2.4GHz operation.

D.11.1 Enhanced Beacon Request and Enhanced Beacon (response)

When networks are required to provide a mechanism for joining devices to sort through multiple PANs in overlapping proximity to find the correct network to join, the Enhanced Beaconing capability introduced in IEEE Std 802.15.4 2015 can be used to accomplish this.

In addition to supporting the existing usage of beaconing for the purpose of surveying IEEE Std 802.15.4 networks, devices operating in the GB Smart Energy and Regional Sub-GHz FSK bands SHALL (for devices operating in the 2.4 GHz band it is optional) support the use of Enhanced Beaconing. Support of Enhanced Beaconing includes supporting the Enhanced Beacon Request for joining and rejoining, and the Enhanced Beacon Filtering and Enhanced Beacon (response) for joining, along with any additions/changes specified in this section.

While it is possible for IEs to be included in any MAC frame, devices SHALL only include IEs in frames during the joining and rejoining process when sending Enhanced Beacon Requests and Enhanced Beacons. IEs SHALL NOT be included and the length of the Information Element field SHALL be 0, as shown in Figure D-13, for any other MAC frame. When using Enhanced Beacon Requests and Enhanced Beacons, the length of the Information Element field SHALL NOT be 0.

The Enhanced Beacon Request (EBR) and Enhanced Beacon (EB) are very similar in format to the Beacon Request and Beacon from prior specifications. The Enhanced types provide a mechanism to reduce the beacon storm typically seen when a non-Enhanced Beacon Request is used. The Enhanced types set the Frame Version field to b010. EBRs are achieved by sending a Beacon Request command in a MAC command frame, with the Frame Version field set to b010. A version of b010 indicates the frames are compatible with IEEE Std 802.15.4 2015 and NOT compatible with IEEE Std 802.15.4-2003 nor compatible with IEEE Std 802.15.4-2006. Devices implemented with earlier versions of the IEEE Std 802.15.4 specification will drop the received frames when the Frame Version field is set to b010. When using EBRs and EBs, devices SHALL set the Frame Version field to b010.

EBs and EBRs enable use of a new field in the IEEE Std 802.15.4 MAC protocol known as Information Elements (IEs). These IEs contain arbitrary data and are used in two ways:

1. The IEs contain arbitrary information that the EBs can advertise to un-joined devices about the MLME parameters or higher level application running on top of the IEEE Std 802.15.4 network.
2. The IEs can contain filter criteria that restrict the types of devices or the number of devices that respond to EBRs.

When a device receives unrecognized IEs, the unrecognized IEs are ignored, and the rest of the frame, including any recognized IEs are processed.

The Enhanced Beacon Request (EBR) along with the Enhanced Beacon (EB) response, as defined below, MAY be used in any band in which Zigbee operates. When EBR and EB are used in systems with legacy devices, the legacy devices will ignore the frames, since EBR and EB are version 2 frames.

D.11.1.1 Enhanced Beacon Filter IE (used for joining)

The EB Filter is described in 7.4.4.6 of IEEE Std 802.15.4-2020. When joining a network a device SHALL send an EBR containing the EB Filter IE and SHALL set the Permit Joining On field of the EB Filter IE to 1. Joining devices MAY additionally set either or both of the “Include Link Quality Filter” and the “Include Percent Filter” fields of the EB Filter IE to 1. Devices will not make use of the Attribute IDs and therefore the Attribute IDs Length field of the EB Filter IE SHALL be set to 0.

When a joining device sends the EBR containing the EB Filter IE, it SHALL set the PAN ID Compression field bit of the Frame Control field to 1. The joining device SHALL set the Destination Addressing Mode field of the Frame Control field to 10 and the Source Addressing Mode field of the Frame Control field to 11. The joining device SHALL set the Destination Address to the broadcast short address (0xffff), the Destination PAN ID to the broadcast PAN ID (0xffff), and the Source Address to the extended address of the joining device. A Source PAN ID is not included.

The format of the frame when an EBR is sent during initial joining, including the TX Power IE defined in D.11.2, SHALL be as defined Figure D-1.

Field	Sub-field	Bytes	Value	Notes
PHY Length		1	36	
MAC Frame Control		2	0xEA43	
	Frame Type (bits 0-2)		3	
	Security Enabled (bit 3)		0	
	Frame Pending (bit 4)		0	
	ACK Required (bit 5)		0	
	PAN ID Compression (bit 6)		1	Formerly known as Intra-PAN
	Reserved (bit 7)			
	Sequence Number Suppression (bit 8)		0	
	IE List Present (bit 9)		0	
	Destination Address Mode (bits 10-11)		2	Short Address Mode
	Frame Version (bits 12-13)		2	Normally Zigbee Uses 0 for the version
	Source Address Mode (bits 14-15)		3	Long Address Mode
Sequence Number		1	0	May be any value
Destination PAN ID		2	0xFFFF	
Destination Address		2	0xFFFF	
Source PAN Identifier		0		Not Present since PAN ID Compression = 1
Source Address		8	0x0123456789ABCDEF	This will be the device's actual EUI64
Auxiliary Security Header		0		Not Present
Header IE Termination		2	0x3F00	Termination IE Payload IE to follow
	Length (bits 0 – 6)		0	Length field does not include frame control
	Group ID (bits 11-14)		1	0x01 = MLME (Nested)
	Type (bit 15)		0	0 = Header IE
Payload IE		2	0x8803	MLME Nested
	Length (bits 0 - 10)		3	Length field does not include Frame Control
	Group ID (bits 11-14)		1	0x01 = MLME (Nested)
	Type (bit 15)		1	1 = Payload IE
Sub-IE Descriptor		2	0x1E01	Enhanced Beacon Filter
	Length		1	Length field does not include Frame Control
	Sub-ID (bits 8-14)		0x1E	0x1E = EB Filter
	Type (bit 15)		0	0 = Short (0-255 length)
	Filter Control Field	1	0x01	Filters Enabled: Permit Joining on

Field	Sub-field	Bytes	Value	Notes
Payload IE		2	0x9006	Vendor Specific: Zigbee OUI
	Length (bits 0-10)		0x06	
	Group ID (bits 11-14)		0x02	Vendor Specific
	Type (bit 15)		1	1 = Payload IE
Vendor OUI		3	0x4A191B	The Connectivity Standards Alliance's assigned CID value is 4A-19-1B
Sub-IE Descriptor		2	0x0041	TX Power
	Length (bits 0-5)		0x01	
	Sub-ID (bits 6-15)		0x0001	Zigbee TX Power IE
	TX Power	1	0x1B	TX Power (dBm) used to send this frame (e.g. 27)
Payload IE		2	0xF800	Termination IE
	Length (bits 0-10)		0	
	Group ID (bits 11-14)		0xF	Payload Termination IE
	Type (bit 15)		1	1 = Payload IE
Command Frame Identifier		1	0x07	0x07 = Beacon Request
CRC		2	0x1234	This value is calculated according to the rest of the frame.

Figure D-1. Enhanced Beacon Request Format

D.11.1.2 Additional Filtering When Joining

Using the EB Filter helps to filter out networks only when a single network within an area is commissioned at a time. However, there are many cases where multiple devices are commissioned to different PANs in close proximity within the same time window. This could occur for example within a multiple dwelling unit where there are unique PANs per apartment. Installers MAY setup multiple networks at the same time, or end users within close proximity are enrolled into a utility program at the same time and are commissioning devices. In these cases simply filtering on open networks will still produce a large number of results.

Unlike Beacon Requests, EBRs MAY contain the source address field and therefore the extended address of the device sending the EBR. This provides a mechanism for an additional layer of filtering to occur. The Trust Center, for the networks capable of doing so, SHALL update the *mibJoiningIeeeList* by adding the IEEE addresses of devices allowed to join the network and sending the updated list to coordinators/routers in the network. The coordinators/routers SHALL store the *mibJoiningIeeeList* within volatile memory. Coordinators/Routers in the network SHALL use the *mibJoiningPolicy* to set their current permitted joining mode. The *mibJoiningIeeeList* and *mibJoiningPolicy* attributes are described in Table D-4.

In the case where the network has no Trust Center (distributed network), routers SHALL NOT support the IEEELIST_JOIN *mibJoiningPolicy*.

Additionally when a new coordinator/router joins the network the Trust Center SHALL send the *mibJoiningIeeeList* to the newly joined coordinator/router. Trust center SHALL use the Mgmt_NWK_IEEE_Joining_List_rsp to inform the devices at the join time of the *mibJoiningIeeeList*.

In case of updates to the *mibJoiningIeeeList* while no new device has joined, the trust center SHALL send Mgmt_NWK_IEEE_Joining_List_rsp to inform the network devices of the latest *mibJoiningIeeeList*.

Table D-4. Joining MAC PIB Attributes

Attribute	Type	Range	Description	Default

Read/Write Attributes				
mibJoiningIeeeList	List of 64bit mac addresses	-	List containing the 64bit *macExtendedAddress* of each device permitted to join.	Empty
mibJoiningPolicy	Enumeration	NO_JOIN, ALL_JOIN, IEEELIST_JOIN	Sets the permitted mode of joining	NO_JOIN
mibIeeeExpiryInterval	Integer	1-1440 (min)	Time before clearing *mibJoiningIeeeList*.	5 (min)
mibIeeeExpiryInterval-Countdown	Integer	0-86400 (sec)	Number of seconds remaining before clearing *mibJoiningIeeeList* and setting *mibJoiningPolicy* to NO_JOIN.	Empty

The combination of the *mibJoiningIeeeList* and *mibJoiningPolicy* SHALL, for the MAC interfaces capable of doing so, be used as part of the filtering and joining process used in response to EBRs as described below. For the MAC interfaces unable to support the *mibJoiningIeeeList* capability only the NO_JOIN and ALL_JOIN options of the *mibJoiningPolicy* SHALL be used.

The higher layer SHALL manage the *mibJoiningIeeeList* via the MLME-GET.request and MLME-SET.request primitives. Whenever the MLME-SET.request is used to modify the *mibJoiningIeeeList* a timer known as *mibIeeeExpiryIntervalCountdown* SHALL be started. The initial value of *mibIeeeExpiryIntervalCountdown* SHALL be set to 60 times the *mibIeeeExpiryInterval*. Every second the *mibIeeeExpiryIntervalCountdown* shall be decremented until it reaches 0. Upon reaching zero, the *mibJoiningIeeeList* SHALL be set to an empty list and the *mibJoiningPolicy* SHALL be set to NO_JOIN.

If a coordinator/router has not been told the *macExtendedAddress* of devices joining into the network, prior to receiving an EBR, then the *mibJoiningIeeeList* in the coordinator/router SHALL remain empty.

Coordinators/Routers hearing an EBR with the EB Filter IE, who have their *mibJoiningPolicy* set to NO_JOIN, SHALL NOT allow devices to join and an EB SHALL NOT be sent.

Coordinators/Routers hearing an EBR with the EB Filter IE, who have their *mibJoiningPolicy* set to ALL_JOIN or IEEELIST_JOIN, SHALL first filter on the EB Filter IE as described previously. If the applied filter indicates that an EB is to be generated, then the coordinator/router SHALL examine their *mibJoiningPolicy:*

- If the *mibJoiningPolicy* is set to ALL_JOIN, then any device MAY join that coordinator/router and an EB SHALL be sent.
- If the *mibJoiningPolicy* is set to IEEELIST_JOIN, then a secondary filter SHALL be applied. This secondary filter SHALL examine the *mibJoiningIeeeList:*
    - If the contents of the *mibJoiningIeeeList* contains an address matching the source address field in the command frame containing the EBR, then an EB SHALL be sent.
    - If the contents of the *mibJoiningIeeeList* does not contain an address matching the source address field in the command frame containing the EBR, then an EB SHALL NOT be sent.

When an EB is sent by a coordinator/router in response to an EBR, and is sent as part of the joining process (i.e. an EBR containing the EB Filter IE), it SHALL set the PAN ID Compression field bit of the Frame Control field to 0. The coordinator/router SHALL set the Destination Addressing Mode field of the Frame Control field to 00 and Source Addressing Mode field of the Frame Control field to 11. By setting the Destination Address Mode to NONE (00), devices from other networks can receive the EB to perform PAN ID conflict detection and resolution the same as it would for a standard (non-enhanced) Beacon.

The coordinator/router SHALL set the Source PAN ID to the *macPanId*, and the Source Address field to the extended address of the device transmitting the beacon. The Destination PAN ID and Destination Address fields are not included.

For consistency the standard beacon information SHALL be presented to the Zigbee layers when receiving an Enhanced Beacon, whether it is during the initial joining or the rejoining process. To facilitate this EBs SHALL include the EB Payload IE. The EB Payload IE is a Zigbee Payload (Nested) IE. The Sub-ID Value for the EB Payload IE is given in .

The Content field format of the EB Payload IE is shown in Figure D-2.

Bits: 0-7	**8-11**	**12-15**	**16-17**	**18**	**19-22**	**23**	**24-87**	**88-111**	**112-119**
Protocol ID	Stack Profile	NWK Protocol Version	Reserved	Router Capacity	Device Depth	End Device Capacity	NWK EXT PAN ID	Tx Offset	NWK Update ID

Bits: 120-123	**124-127**	**128-131**	**132**	**133**	**134**	**135**	**136-151**
Beacon Order	Super-Frame Order	Final CAP Slot	Battery Life Extension	Reserved	PAN Coordinator	Association Permit	Source Short Address

Figure D-2. EB Payload IE Content Field Format

An EB frame sent in response to an EBR received during initial joining, including the TX Power IE defined in D.11.2, SHALL be formatted as defined in Figure D-3.

Field	Bytes	Value	Notes
PHY Length	1	49	
Frame Control	2	0xE200	
Frame Type (bits 0-2)		0	Beacon Frame
Security Enabled (bit 3)		0	
Frame Pending (bit 4)		0	
ACK Required (Bit 5)		0	
PAN ID Compression (bit 6)		0	Formerly known as Intra-PAN
Reserved (bit 7)		0	
Sequence Number Suppression (bit 8)		0	
IE List Present (bit 9)		1	IEs are present
Destination Address Mode (bits 10-11)		0	None Address Mode (required for PAN ID conflict detection)
Frame Version (bits 12-13)		2	Version 2 = Enhanced Beacon
Source Address Mode (bits 14-15)		3	Long Address mode. (Short address in EB Payload IE)
Sequence Number	1	0x00	0-255
Destination PAN ID	0		Not present since PAN ID Compression = 0
Destination Address	0		Not present
Source PAN Identifier	2	0x1234	Sender's macPANid
Source Address	8	0x0123456789ABCDEF	Sender's Address
Auxiliary Security Header	0		Not Present since Security Enabled = 0
HIE Termination (Payload IE to follow)	2	0x3F00	Termination IE
Length (bits 0 - 6)		0	Length field does not include Frame control
Element ID (bits 7 - 14)		0x7E	Header Termination IE 1
Type (bit 15)		0	Header IE = 0
PIE (Vendor Specific: ZigBee OUI)	2	0x901B	
Length (bits 0 - 10)		0x1B	27
Group ID (bits 11 - 14)		0x2	Vendor Specific
Type (bit 15)		1	1 = Payload IE
Vendor OUI	3	0x4A191B	The ZigBee Alliance's assigned CID value is 4A-19-1B
Sub-IE descriptor (EB Payload)	2	0x0093	
Length (bits 0 – 5)		0x13	19
Sub-ID (bits 6 -15)		0x0002	ZigBee EB Payload IE
Beacon Payload	15		Standard ZigBee PRO Beacon Payload
Protocol ID (bits 0-7)		0x00	
Stack Profile (bits 8-11)		0x2	
NWK Protocol Version (bits 12-15)		0x2	
Reserved (bits 16-17)		0	
Router Capacity (bit 18)		1	0 or 1
Device Depth (bits 19-22)		0x0	0 to 15
End Device Capacity (bit 23)		1	0 or 1
NWK EXT PAN ID (bits 24-87)		0x1122334455667788	EXT PAN ID of network
Tx Offset (bits 88-111)		0xFFFFFF	
NWKUpdate ID (bits 112-119)		0x00	0-255
Superframe Specification	2	0xCFFF	
Beacon Order (bits 0-3)		0xF	
SuperFrame Order (bits 4-7)		0xF	
Final CAP Slot (bits 8-11)		0xF	
Battery Life Extension (bit 12)		0	0 or 1
Reserved (bit 13)		0	
PAN Cordinator (bit 14)		1	0 or 1
Association Permit (bit 15)		1	0 or 1
Source Short Address (bit 0-15)	2	0x1234	Short address of device sending EB
Sub-IE descriptor (TX Power)	2	0x0041	
Length (bits 0 – 5)		0x01	
Sub-ID (bits 6 -15)		0x0001	ZigBee TX Power IE
TX Power	1	0x1B	TX Power (dBm) used to send this frame (e.g. 27).
PIE (Termination)	2	0xF800	Termination IE
Length (bits 0 - 10)		0	
Group ID (bits 11 - 14)		0xF	Payload Termination IE
Type (bit 15)		1	
CRC	2	0x1234	This value is calculated according to the rest of the frame.

Figure D-3.

After successful joining of a device it is EXPECTED that the Trust Center can update the *mibJoiningIeeeList* to remove the IEEE addresses of devices that have successfully joined and send the updated list to coordinators/routers in the network. This MAY be done via sending the ZDO message Mgmt_NWK_IEEE_Joining_List_rsp. The ZDO layer on the coordinator/router receiving this message in turn will inform the MLME.

The *mibJoiningPolicy* only affects when Enhanced Beacons are sent. It has no relation to *macAssociationPermit* within the MLME. It is EXPECTED that the higher layer will inform the MLME separately of the state of *macAssociationPermit*, *mibJoiningPolicy*, *mibJoiningIeeeList*, *mibIeeeExpiryInterval*, and *mibIeeeExpiryIntervalCountdown*. These MLME attributes can be uniformly managed across the Zigbee Network by the ZDO.

	Response When		
	Permit Join true	*mibJoiningPolicy is IEEELIST_JOIN and IEEE found*	*Permit Join false*
Association Request	Success	Success	Reject
Beacon Request	Beacon with pjoin=true		Beacon* with pjoin=false
Enhanced Beacon Request	respond per filter	respond per filter	no response

* If *mibJoiningPolicy* is IEEELIST_JOIN a Beacon Request is always responded to with a Beacon with pjoin=true.

D.11.1.3 Rejoining a Network

It is possible in any Zigbee network that the network MAY be moved to a different channel or short PAN ID due to contention. Channel changes MAY be more common in the sub-GHz bands vs. that of 2.4 GHz, due to the fact that some channels MAY be limited in their allowable power levels as a result of regional regulatory restrictions. Channel changes can take place temporarily for commissioning or they could be a permanent change to accommodate a new device that needs to use a higher power level in order to successfully communicate with the coordinator/router.

This change in channel or short PAN ID will cause any device that misses the notification (such as a sleepy end device) to perform a rejoin. A rejoin requires an EBR and response (EB) very similar to joining.

Rejoining devices will want to be able to use a similar method to joining devices to filter out extraneous beacons of other networks. However, they will want to filter based on the Extended PAN ID, which is a universal identifier for the specific network they were already commissioned on. They do not want to filter based on the permit joining flag since that flag is only used by new devices when first joining the network.

When rejoining a network a device SHALL use the EBR, but it SHALL NOT contain the EB Filter IE. The rejoining device SHALL include the nested Zigbee Payload IE, containing the Rejoin IE and TX Power IE.

The rejoining device SHALL set the Network Extended PAN ID field of the Rejoin IE to the *nwkExtendedPANId* of the NIB of the rejoining device.

When a rejoining device sends an EBR, it SHALL set the PAN ID Compression field bit of the Frame Control field to 1. The rejoining device SHALL set the Destination Addressing Mode field of the Frame Control field to 10 and the Source Addressing Mode field of the Frame Control field to 11. The rejoining device SHALL set the Destination Address to the broadcast short address (0xffff), the Destination PAN ID to the broadcast PAN ID (0xffff), and the Source Address to the extended address of the rejoining device. A Source PAN ID is not included.

The format of the frame when an EBR is sent during rejoining SHALL be as defined in Figure D-4.

Field	Bytes	Value	Notes
PHY Length	1	43	
Frame Control	2	0xEA43	According to 802.15.4e the EBR is a Beacon Request with version b010
Frame Type (bits 0-2)		3	MAC Command
Security Enabled (bit 3)		0	
Frame Pending (bit 4)		0	
ACK Required (Bit 5)		0	
PAN ID Compression (bit 6)		1	Formerly known as Intra-PAN
Reserved (bit 7)		0	
Sequence Number Suppression (bit 8)		0	
IE List Present (bit 9)		1	
Destination Address Mode (bits 10-11)		2	Short Address mode.
Frame Version (bits 12-13)		2	Normally Zigbee uses 0 for the version
Source Address Mode (bits 14-15)		3	Long Address mode.
Sequence Number	1	0	This may be any value.
Destination PAN ID	2	0xFFFF	Broadcast
Destination Address	2	0xFFFF	Broadcast
Source PAN Identifier	0		Not present since PAN ID Compression = 1
Source Address	8	0x0123456789ABCDEF	This will be set to the device's actual EUI64
Auxiliary Security Header	0		Not Present since Security Enabled = 0
HIE Termination (Payload IE to follow)	2	0x3F00	Termination IE
Length (bits 0 - 6)		0	Length field does not include Frame control
Element ID (bits 7 - 14)		0x7E	Header Termination IE 1
Type (bit 15)		0	Header IE = 0
PIE (Vendor Specific: ZigBee OUI)	2	0x9012	
Length (bits 0 - 10)		0x12	18
Group ID (bits 11 - 14)		0x2	Vendor Specific
Type (bit 15)		1	1 = Payload IE
Vendor OUI	3	0x4A191B	The ZigBee Alliance's assigned CID value is 4A-19-1B
Sub-IE descriptor (Rejoin)	2	0x000A	
Length (bits 0 – 5)		0x0A	
Sub-ID (bits 6 -15)		0x0000	ZigBee Rejoin IE
Extended PAN ID	8	0x1234567890abcdef	Extended PAN ID of Network to Rejoin
Source Short Address (bit 0-15)	2	0x1234	Short address of device sending EBR
Sub-IE descriptor (TX Power)	2	0x0041	
Length (bits 0 – 5)		0x01	
Sub-ID (bits 6 -15)		0x0001	ZigBee TX Power IE
TX Power	1	0x1B	TX Power (dBm) used to send this frame (e.g. 27).
PIE (Termination)	2	0xF800	Termination IE
Length (bits 0 - 10)		0	
Group ID (bits 11 - 14)		0xF	Payload Termination IE
Type (bit 15)		1	
Command Frame Identifier	1	0x07	0x07 = Beacon Request.
CRC	2	0x1234	This value is calculated according to the rest of the frame.

Figure D-4. Enhanced Beacon Frame Format

Coordinators/Routers hearing an EBR with a nested Zigbee Payload IE, containing the Rejoin IE and TX Power IE, SHALL filter on the Network Extended PAN ID field of the Rejoin IE. If the Network Extended PAN ID field of the Zigbee IE matches the value of the *nwkExtendedPanID* in their NIB, then the coordinator/router SHALL send an EB. If the Network Extended PAN ID field of the Zigbee IE does not match the value of the *nwkExtendedPanID* in their NIB, then the coordinator/router SHALL NOT send an EB.

The EB frame sent in response to an EBR received during rejoining SHALL be the same format as the EB sent for initial joining.

D.11.1.4 Zigbee Payload IE

The general format of a Payload IE is given in the IEEE Std 802.15.4-2020 Standard.

The Zigbee Payload IE is a Vendor Specific Payload IE (Group ID = 0x2) using the Zigbee OUI value of 0x4A191B.

The Zigbee Payload (Nested) IE SHALL be formatted as shown in Figure D-5.

Bits: 0-5	**6-15**	**Octets: Variable**
Length	Sub-ID	Content

Figure D-5. Zigbee Payload Nested IE

The Sub-ID field values for Zigbee Payload Nested IEs are shown in Table D-5.

Table D-5. Sub-ID Allocation for Zigbee Payload Nested IE

Sub-ID value	Name
0x00	Rejoin IE
0x01	TX Power IE
0x02	EB Payload IE
0x003-0x3ff	Reserved

The Rejoin IE Content field SHALL be formatted as illustrated in Figure D-6.

Octets: 8	**2**
Network Extended PAN ID	Sender Short Address

Figure D-6. Rejoin IE Content Field Format

The TX Power IE Content field SHALL be formatted as illustrated in Figure D-7.

Octets: 1
TX Power (in dBm - used to send the frame)

Figure D-7. TX Power IE Content Field Format

The EB Payload IE Content field SHALL be formatted as illustrated in Figure D-8. Refer to 7.4.4.6 of IEEE Std 802.15.4-2020 for more detailed information.

Octets: 15	2	2
Beacon Payload	Superframe Specification	Sender Short Address

Figure D-8. EB Payload IE Content Field Format

D.11.1.5 Support for Non-Enhanced Beaconing

Devices supporting Enhanced Beaconing SHALL also support the usage reception of non-enhanced Beaconing for the purposes of surveying IEEE Std 802.15.4 networks.

- All coordinators/routers SHALL process Beacons for PAN ID Conflict, as per the standard mechanism.
- All coordinators/routers SHALL respond to Beacon Requests with a Beacon.

o When power control is used the Beacon SHALL be sent at a power level that corresponds to the highest power level of all the known devices joined to the coordinator/router plus 6 dB, up to the maximum allowable for the channel.

Example: Four devices are joined to a Sub-GHz coordinator/router. One device requires that the coordinator/router transmit at a power level of +10 dBm while all others require less. If the channel that the Beacon Request was received on had a maximum allowable transmit power of +27 dBm, the coordinator/router would respond to the Beacon Request using a transmit power level of +16 dBm. If however, the channel that the Beacon Request was received on had a maximum allowable transmit power of +14dB, the coordinator/router would respond to the Beacon Request using a transmit power level of +14 dBm.

o When power control is not used the Beacon SHALL be sent at the power level currently being used for the channel.

D.11.1.6 Association Following an Enhanced Beacon

To remain consistent with the IEEE Std 802.15.4-2020 specification a device sending an Enhanced Beacon request SHALL use long destination address mode when sending the Association Request. However devices receiving the Association request SHALL accept either long or short destination address modes. The receiving destination SHALL NOT filter the association request based on a prior beacon request.

D.11.2 MAC Support for Power Control

D.11.2.1 Power Control Primitive Parameters

D.11.2.1.1 RSSI Parameter

The RSSI of the received packet SHALL be measured for each received packet. This measurement MAY be taken over any portion of the received frame. Upon reception of a packet the RSSI (in dBm) of the received packet SHALL be passed up to the MAC using the appropriate MLME primitive. RSSI is an 8-bit signed integer representing the measured receive power dBm, in one dB increments. To accomplish this the RSSI parameter SHALL be added as follows:

Name	Type	Valid Range	Description
RSSI	Signed integer	See Table D-14.	The Received Signal Strength Indicator is a measure of the RF power level (in dBm) at the input of the transceiver measured during any portion of the frame being received.

D.11.2.2 TX Power IE for EBR and EB (During Joining and Rejoining Process)

To facilitate power control during the Joining and Rejoining process, the Zigbee Payload (Nested) IE, shown in Figure D-8, SHALL be sent when using the EBR and EB. The Zigbee Payload (Nested) IE SHALL contain the TX Power IE. The Content field format of the TX Power IE is shown in Figure D-7. The Sub-ID Value for the TX Power IE is given in Table D-5. The TX Power field SHALL be set to the transmit power setting of the device sending the packet. TX Power is an 8-bit signed integer representing the TX Power in dBm, in one dB increments.

D.11.2.3 Power Control Information Table

To facilitate power control in the MAC, used for standard messages as well as for acknowledgements (ACKs), the MAC SHALL maintain a Power Control Information Table. The table SHALL contain the links presently being used by the MAC. These include links that are being established as well as links that have been established and have an entry in the neighbor table. The table SHALL NOT persist over a power reset/interruption. After a power reset/interruption devices SHALL start at their maximum power level (i.e. up to and including +14 dBm for GB 868) and renegotiate down from there. The format of the information stored for each link represented in the Power Control Information Table is shown in Figure D-9.

Octets	2	8	1	1	1
Data Type	Short Address	IEEE Address	TX Power Level	Last RSSI Level	NWK Negotiated

Figure D-9. Format of Power Control Information Table Entry

The Power Control Information Table SHALL support at least 1 entry.

Both the Short Address and the IEEE Address are required in the EB during joining and rejoining. As soon as an EB is received an entry is created in the MAC Power Control Information Table. Even though the use of the short and long addresses MAY allow for the required Tx power for ACKs to be set quickly - it MAY still be extremely tight timing to perform a lookup. Therefore ACKs MAY be sent with a Tx power equal to the maximum Tx Power Level in the Power Control Information Table if there is not enough time to perform a lookup in the MAC Power Control Information Table.

If the sending device has an entry in the MAC Power Control Information Table then this entry is used as the power level for all transmissions to this device. If the sending device has no entry then the maximum power level for the channel is to be used.

During the joining / rejoining process the entry in the table remains and the entry NWK Negotiated remains 0. Should the device successfully join and an entry is created in the neighbor table then the flag NWK Negotiated SHOULD be set to 1.

A regular NWK housekeeping routine (used for power negotiation) checks for table entries not having a NWK Negotiated flag set to 1, and if after 10 seconds from an entry being made during the joining / rejoining process the NWK Negotiated flag has not been set to 1, the entry SHALL be removed from the Power Control Information Table.

D.11.2.3.1 MLME-GET-POWER-INFORMATION-TABLE.request

The MLME-GET-POWER-INFORMATION-TABLE.request primitive returns the Power Control Information entry for the link pair. The request MAY include the Short Address and/or the IEEE (Long) Address.

D.11.2.3.1.1 Semantics of the Service Primitive

The semantics of the MLME-GET-POWER-INFORMATION-TABLE.request primitive are as follows:

```
MLME-GET-POWER-INFORMATION-TABLE.request    {
                                            Short Address
                                            IEEE Address
                                            }
```


Name	Type	Valid Range	Description
Short Address	16bit mac address	0-0xfff7	Short address of the link pair to transmit the packet to.
IEEE Address	64bit mac address	valid mac address	Extended (IEEE) address of the link pair to transmit the packet to.

D.11.2.3.2 MLME-GET-POWER-INFORMATION-TABLE.confirm

The MLME-GET-POWER-INFORMATION-TABLE.confirm primitive returns the status and the information requested by the MLME-GET-POWER-INFORMATION-TABLE.request.

D.11.2.3.2.1 Semantics of the Service Primitive

The semantics of the MLME-GET-POWER-INFORMATION-TABLE.confirm primitive are as follows:

```
MLME-GET-POWER-INFORMATION-TABLE.confrm    {
                                           Status
                                           Elements in Figure D-9
                                           }
```

Name	Type	Valid Range	Description
Status	Enumeration	SUCCESS, FAIL, UNSUPPORTED	Used to indicate if an entry was found for the pair requested.
Elements in Figure D-9	Variable	-	Elements in Figure D-9 for the pair requested.

D.11.2.3.3 MLME-GET-POWER-INFORMATION-TABLE.request

The MLME-GET-POWER-INFORMATION-TABLE.request primitive adds the Power Control Information entry for the link pair.

D.11.2.3.3.1 Semantics of the Service Primitive

The semantics of the MLME-GET-POWER-INFORMATION-TABLE.request primitive are as follows:

```
MLME-GET-POWER-INFORMATION-TABLE.request   {
                                           Elements in Figure D-9
                                           }
```

Name	Type	Valid Range	Description
Elements in Figure D-9	Variable	-	Elements in Figure D-9 for the pair requested.

D.11.2.3.4 MLME-SET-POWER-INFORMATION-TABLE.confirm

The MLME-SET-POWER-INFORMATION-TABLE.confirm primitive returns the status and the information requested by the MLME-SET-POWER-INFORMATION-TABLE.request.

D.11.2.3.4.1 Semantics of the Service Primitive

The semantics of the MLME-SET-POWER-INFORMATION-TABLE.confirm primitive are as follows:

```
MLME-SET-POWER-INFORMATION-TABLE.confrm        {
                                               Status
                                               }
```

Name	Type	Valid Range	Description
Status	Enumeration	SUCCESS, FAIL, UNSUPPORTED	Used to indicate if an entry for the pair was successfully added to the Power Information Table.

D.11.2.4 Power Control Functional Description

D.11.2.4.1 Power Control during Energy Scans and Active Scans

The Tx power level used during Energy scans SHALL be at whatever power level the devices are currently transmitting at.

The Tx power level used during Active scans (when sending beacon requests and not enhanced beacon requests) during network formation SHALL be at the device's maximum power level (i.e. up to and including +14 dBm for GB 868) and the resulting beacons SHALL be sent at the device's maximum power level (i.e. up to and including +14 dBm for GB 868).

D.11.2.4.2 Power Control during EBR and EB Frames

The use of the EBR and the EB Frame with embedded TX Power IE allows a quick power level negotiation before the bulk of the communication has started. This ensures that the network traffic, especially in the early stages of joining a Zigbee network will be limited in range due to the power level negotiation, thus reducing the interference of the Zigbee network to other Zigbee networks and to other users on these frequencies.

The initial power level negotiation ensures that the optimum received signal level is maintained at the receiver for good reception. The functional description of initial power negotiation occurring when using an EBR and EB during initial joining is explained below.

- The device SHALL send the initial EBR at it's maximum power for the selected channel.
- The initial EBR contains a TX power IE with the transmit power of the frame
- On receipt of the EBR the normal filtering is carried out as well as the extraction of the TX power IE for further processing: EBRpwr (dBm)
- On receipt of the EBR the RSSI of the Frame is noted: EBRRSSI (dBm)*

  * where the EBRRSSI measurement is in reference to the antenna (i.e. not between the chip and an external LNA)
- The optimum received signal level is defined as OPTRSSI (dBm): defined as 20dB above the sensitivity requirement
- The recipient then calculates the difference between the incoming RSSI level of the EBR and the power level of the EBR to give the effective path loss of this link:

  PATHLOSSpwr (dB) = EBRRSSI (dBm)- EBRpwr (dBm)
- Nonlinear absorption of the packet energy is neglected for the link and it is assumed that the link loss between antennae is symmetric
- In order to establish the optimum power level at the sender, the recipient calculates what power is necessary to overcome this path loss. This is then set as the TX power used for the EB: EBPWR (dBm). This is calculated as follows:

EBPWR (dBm) = OPTRSSI (dBm) - PATHLOSSpwr (dB)

- This transmit power setting is then used AND is coded into the TX Power IE of the EB
- On receipt of the EB in response to a EBR the TX Power IE is read and is used as the power setting for this link

After the initial power control negotiation the TX power values used for the link remain unchanged until the network layer carries out regular housekeeping and can adjust the power levels of the link nodes.

D.11.2.4.3 Ongoing Power Control

Ongoing Power Control is accomplished via. mechanisms implemented in the network layer and SHALL make use of the Power Control Information Table .

D.11.2.4.4 Power Control Tx Power Limits

When implementing power control the Tx power of a device SHALL not be adjusted to exceed the Maximum Transmit Power and Minimum Transmit Power limits defined in D.12.2.2.3.1 and D.12.2.2.3.2 respectively.

D.11.2.4.5 Tx Power for Devices not in Power Control Information Table

Scenarios exist, outside of the EBR/EB exchange for joining and rejoining, where a device transmits to another device that is not in its Power Control Information Table. In these scenarios the transmitting device SHALL transmit at a power equal to the maximum power level for the selected channel.

D.12 GB Smart Energy Sub-GHz FSK PHY Specification

D.12.1 GB Smart Energy Sub-GHz FSK Frame Format

European Sub-GHz FSK SHALL use a 802.15.4 frame formatted in the following manner.

D.12.1.1 PPDU Format for GB Smart Energy Sub-GHz FSK

European Sub-GHz FSK SHALL use the PPDU as defined in 802.15.4, shown in Figure D-10.

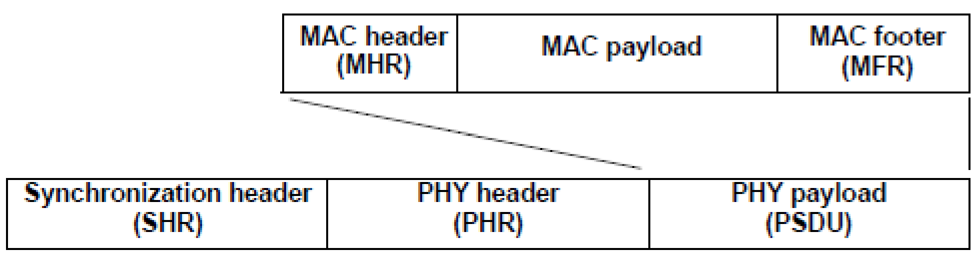


Figure D-10. PPDU

D.12.1.1.1 SHR for GB Smart Energy Sub-GHz FSK

GB Smart Energy Sub-GHz FSK SHALL use the IEEE Std 802.15.4-2020 SUN FSK SHR, as shown in Figure D-11.

Octets: 8	2
Preamble	**SFD**

Figure D-11. SHR

The Preamble field SHALL contain *phyFSKPreambleLength* = 8 multiples of the 8-bit sequence “01010101”, resulting in the following value for the Preamble field:

Preamble: 0101010101010101010101010101010101010101010101010101010101010101

The SFD SHALL be as defined in 20.2.1.2 of IEEE Std 802.15.4-2020 with the following constraint: All devices SHALL support *phyMRFSKSFD* = 0 for uncoded PHR + PSDU fields, resulting in the following value for the SFD field:

SFD: 1001 0000 0100 1110

When *phyMRFSKSFD* = 0, FEC is not supported. *phyMRFSKSFD* = 1 is not supported.

D.12.1.1.2 PHR for GB Smart Energy Sub-GHz FSK

European Sub-GHz FSK SHALL use the IEEE Std 802.15.4-2020 SUN FSK PHR, with the following settings as shown in Figure D-12.

Bit string index	**0**	**1-2**	**3**	**4**	**5-15**
Bit settings	0	00	1	1	00001111111
Field name	Mode Switch	Reserved	FCS Type	Data Whitening	Frame Length*

Figure D-12. PHR

* Maximum value of Frame Length

where:

- The Mode Switch bit SHALL be set to 0 (signifying that only a single data rate is used and that mode switching is not supported);
- The Reserved bits SHALL be set to 0;
- The FCS Type bit SHALL be set to 1 (signifying that a 2 byte FCS is used - to align with the 2 byte FCS used by Zigbee for the 2.4 GHz PHY);
- The Data Whitening bit SHALL be set to 1 (signifying data whitening of the PSDU is always enabled);
- The Frame Length field SHALL indicate the total number of octets contained in the PSDU (i.e., PHY payload). The PHY constant *aMaxPHYPacketSize* SHALL be set to 127, and therefore the maximum value the Frame Length field SHALL be 127. This is done to align with the maximum Frame Length used by Zigbee for the 2.4 GHz PHY.

D.12.1.1.3 PSDU Format for GB Smart Energy Sub-GHz FSK

European Sub-GHz FSK SHALL use the IEEE Std 802.15.4-2020 PSDU for General MAC Frames, with permissible field lengths as shown in Figure D-13, so that it matches the PSDU for General MAC Frames used by Zigbee for the 2.4 GHz PHY.

<table>
<tr><th>Octets: 2</th><th>1</th><th>0/2</th><th>0/2/8</th><th>0/2</th><th>0/2/8</th><th>variable</th><th colspan="2">variable</th><th>variable</th><th>2</th></tr>
<tr><td rowspan="2">Frame Control</td><td rowspan="2">Sequence Number</td><td>Destination PAN Identifier</td><td>Destination Address</td><td>Source PAN Identifier</td><td>Source Address</td><td rowspan="2">Auxiliary Security Header</td><td colspan="2">Information Elements</td><td rowspan="2">Frame Payload</td><td rowspan="2">FCS</td></tr>
<tr><td colspan="4">Addressing fields</td><td>Header IEs</td><td>Payload IEs</td></tr>
<tr><td colspan="8">MHR</td><td colspan="2">MAC Payload</td><td>MFR</td></tr>
</table>

Figure D-13. PSDU for General MAC Frames

While it is possible for IEs to be included in any MAC frame, IEs SHALL only be included in EB frames and MAC Command Frames sending an EBR. Otherwise IEs SHALL NOT be included and the length of the Information Element field SHALL be 0, as shown in Figure D-12.

D.12.2 GB Smart Energy Sub-GHz FSK PHY

D.12.2.1 Modulation Specification

European Sub-GHz FSK SHALL NOT use any of the mandatory or optional modes defined in IEEE Std 802.15.4-2020 SUN FSK. European Sub-GHz FSK SHALL use the modulation requirements as specified in Table D-6.

Table D-6. Modulation Requirements

Parameter	Configuration
Modulation	2-Level GFSK
Data Rate	100 kbps
Tx Filter BT	0.5 (Gaussian)
Modulation Index	0.7

No other modulation parameters are supported.

D.12.2.1.1 Forward Error Correction (FEC)

GB Smart Energy Sub-GHz FSK devices SHALL NOT support FEC coding.

D.12.2.1.2 Data Whitening

GB Smart Energy Sub-GHz FSK devices SHALL support Data whitening as specified in 20.4 of IEEE Std 802.15.4-2020.

D.12.2.1.3 Channels and Frequencies

GB Smart Energy Sub-GHz FSK devices SHALL be capable of operating on all the channels specified, in all of the bands specified in Table D-7.

Table D-7. Total Number of Channels and First Channel Center Frequencies

Frequency band (MHz)	ChanSpacing (MHz)	TotalNumChan	ChanCenterFreq0
863 – 876 (Europe)	0.2	63	863.25
915 – 921 (Europe)	0.2	27	915.35

The exact Channel Plan used in a GB Smart Energy Sub-GHz FSK deployment SHALL be specified using a channel mask.

D.12.2.1.4 Channel Numbering

Channel numbers are assigned as follows:

$$ChanCenterFreq = ChanCenterFreq0 + NumChan * ChanSpacing$$

where *ChanCenterFreq0* is the first channel center frequency in MHz, *ChanSpacing* is the separation between adjacent channels in MHz, and *NumChan* is the channel number from 0 to *TotalNumChan–1*.

For the 863 MHz - 876 MHz band

TotalNumChan = 63

ChanSpacing = 0.2

NumChan goes from 0 to 62

ChanCenterFreq0 = 863.25

This results in the following channel numbers and center frequencies for 863 MHz to 876 MHz shown in Table D-8 which SHALL be used for the 863 MHz - 876 MHz band:

Table D-8. Channels and Center Frequencies for 863 MHz - 876 MHz

Channel #	Fc (MHz)
0	863.25
1	863.45
…	…
61	875.45
62	875.65

For the 915 MHz - 921 MHz band

TotalNumChan = 27

ChanSpacing = 0.2

NumChan goes from 0 to 26

ChanCenterFreq0 = 915.35

This results in the following channel numbers and center frequencies for 915 MHz to 921 MHz shown in Table D-9, which SHALL be used for the 915 MHz - 921 MHz band:

Table D-9. Channels and Center Frequencies for 915 MHz - 921 MHz

Channel #	Fc (MHz)
0	915.35
1	915.55
…	…
25	920.35
26	920.55

D.12.2.1.5 Channel Pages

Four new channel pages are allocated, for GB Smart Energy Sub-GHz FSK, using the existing 32-bit mechanism (5-bits of channel-page, 27-bits of channel-mask), spreading the channels across the 4 pages as follows.

Channel Page	Description
28	863 MHz band, channels 0-26
29	863 MHz band, channels 27-34, 62
30	863 MHz band, channels 35-61 -- the "high-power band"
31	915 MHz band, channels 0-26

D.12.2.2 GB Smart Energy Sub-GHz FSK RF Requirements

D.12.2.2.1 Receiver Requirements

D.12.2.2.1.1 Standard Measurement Conditions

The Standard Measurement Conditions for all receiver requirements SHALL be:

- 139 byte packets (8 bytes Preamble + 2 byte SFD + 2 byte PHR + 127 byte PSDU)
- Receiver power measurements made at the antenna connector
- Packet Error Rate of less than 1%

The PHY RF requirements, when measured under these conditions, are intended to apply to a typical device rather than the worst sample of a batch.

D.12.2.2.1.2 Sensitivity Requirement

Under the Standard Measurement Conditions, GB Smart Energy Sub-GHz FSK devices SHALL meet a Reference Sensitivity as specified in Table D-10.

Table D-10. Receiver Reference Sensitivity Requirement

Reference Sensitivity (dBm)
-99

D.12.2.2.1.3 Co-Channel Rejection Requirement

Under the Standard Measurement Conditions, with the wanted signal at 20 dB above the Reference Sensitivity level in Table D-10 and with an interfering signal modulated with the same modulation as the wanted signal and on the same channel, GB Smart Energy Sub-GHz FSK devices SHALL meet the co-channel rejection requirement at the level relative to the wanted signal level as specified in Table D-11.

Table D-11. Receiver Co-Channel Rejection Requirement

Co-Channel Rejection (dB)
-15

D.12.2.2.1.4 Selectivity Requirements

Under the Standard Measurement Conditions, with the wanted signal at 3 dB above the Reference Sensitivity level in and with an interfering signal modulated with the same modulation as the wanted signal; at the frequency offsets specified in Table D-12, GB Smart Energy Sub-GHz FSK devices SHALL meet the selectivity requirements at a power level equal to the wanted signal plus the Level of Interferer as specified in Table D-12.

Table D-12. Receiver Selectivity Requirements

Frequency Offset (kHz)	Level of Interferer Relative to Wanted (dB)
200	17
400	30
1000	35

D.12.2.2.1.5 Blocking Requirements

Under the Standard Measurement Conditions, with the wanted signal at 3 dB above the Reference Sensitivity level in Table D-10 and with an unmodulated interfering signal at the specified frequency offsets specified in Table D-13, GB Smart Energy Sub-GHz FSK devices SHALL meet the blocking requirements at a power level equal to the wanted signal plus the Level of Interferer as specified in Table D-13.

Table D-13. Receiver Blocking Requirements

Center Frequency of Wanted Signal (MHz)	Frequency Offset of Interferer (MHz)	Level of Interferer Relative to Wanted (dB)
868.05	2	40
868.05	6	45
868.05	10	50

D.12.2.2.2 Receive Power Level (RSSI)

The receiver SHALL be capable of measuring the received power level (reported as the RSSI) of a packet (measured over any portion of the received packet), on a packet by packet basis over at least the range defined in Table D-14.

Table D-14. Receive Power Measurement Range

Receive Power Measurement Lower Minimum	Receive Power Measurement Upper Minimum
-97 dBm	-50 dBm

The receiver SHALL be capable of measuring the received power with the step size and accuracy defined in Table D-15.

Table D-15. Receive Power Measurement Step Size and Accuracy

Receive Power Measurement Step Size	Receive Power Measurement Accuracy
≤ 2 dB	≤ 3 dB

D.12.2.2.3 Transmitter Requirements

D.12.2.2.3.1 Maximum Transmit Power

The Regulated Maximum Transmit Allowable Power for GB Smart Energy Sub-GHz FSK devices, in each of the channels, is as specified in Table D-16.

Table D-16. Regulated Maximum Allowable Transmit Power

IEEE Band (MHz)	Regulated Maximum Transmit Power (dBm)	Channel Number
863-876	14	0-34
863-876	27*	35-61
915-921	14	0-26

* For the Great Britain Sub-GHz FSK Deployment the Maximum Transmit Power for all channels, per. requirement from DECC, SHALL be +14dBm.

D.12.2.2.3.2 Minimum Transmit Power

The Minimum Transmit Power for GB Smart Energy Sub-GHz FSK devices, for all the channels, SHALL be as specified in Table D-17.

Table D-17. Minimum Transmit Power

Minimum Transmit Power
-15 dBm

D.12.2.2.3.3 Transmit Power Step Size and Accuracy

The transmitter SHALL be capable of adjusting its transmit power over the range from the Minimum Transmit Power to the maximum power of the device or the maximum specified power, whichever is reached first, with the step size and accuracy specified in Table D-18.

Table D-18. Transmit Power Step Size and Accuracy

Transmit Power Step Size	Transmit Power Accuracy
$\leq$ 2 dB	$\leq$ 3 dB

D.12.3 Channel Plan and Masks

Given the PHY requirements above a total of 90 possible channels are available for GB Smart Energy Sub-GHZ FSK deployment. However, depending on regional restrictions and/or considerations due to ER-GSM, alarm channels, etc. the number of usable channels for any specific country will be less and a mask SHALL be applied.

Due to interferers or restricted use in each of these bands Great Britain Sub-GHz FSK SHALL mask out the channels as indicated below in Table D-19.

Table D-19. Great Britain Sub-GHz FSK Channel Plan and Mask

	Channels Available for GB Smart Energy Sub-GHz FSK Deployment				Channel Mask for GB Smart Energy Sub-GHz FSK Deployment			
			# Avail. Channels					
Channel Page	**Band (channel numbers)**	**Max. Power**	**Example 1 all @ +14 dBm**	**Example 2 all @ Max Pwr**	**Channels #'s to Mask Out**	**# Useable Channels**	**Start Fc* (MHz)**	**End Fc* (MHz)**
28	863-868 MHz band (channels 0-26)	+14 dBm	27	27	None	27 (0-26)	863.25	868.45
29	868-870, 870-876 MHz band (channels 27-34, 62)**	+14 dBm	9	9	62	8 (27-34)	868.65	870.05
30	870-876 MHz band (channels 35-61)	+14 dBm	27	0	49-61	14 (35-48)	870.25	872.85
		or						
		+27 dBm	0	27	-	-	-	-
31	915-921 MHz band (channels 0-26)	+14 dBm	27	27	13-26	13 (0-12)	915.35	917.75
	Total # of Channels @ +14 dBm		90	63		62		
	Total # of Channels @ +27 dBm		0	27		0		
	Total # of All Channels		**90**	**90**		**62**		

*Fc spacing = 200 kHz

**While the alarm channels in this band MAY be restricted in other European countries, and therefore SHOULD be masked out, these channels are approved for use in UK per IR 2030 and EN 300-220 [B12].

For example, the Channel Mask for GB Smart Energy Sub-GHz FSK Deployment shown in results in the following channel page bit mask representations.

Channel Page	Bit Mask Representation
28	1 1 1 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
29	1 1 1 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1
30	1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1
31	1 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1

D.13 GB Smart Energy and Regional PHY Sub-GHz FSK MAC Specification

The GB Smart Energy Sub-GHz FSK MAC SHALL utilize the same MAC used by the Zigbee PRO Network Stack. However, in order to meet the added requirements for sub-GHz use in the UK, several additional MAC capabilities SHALL also be supported. The functions of the additional capabilities are included below, along with the descriptions and requirements of each of the capabilities.

D.13.1 MAC Support for Duty Cycle Monitoring

D.13.1.1 Duty Cycle Monitoring Specific MAC Constants and PIB Attributes

The MAC Sublayer Constants required to support Duty Cycle Monitoring are given in Table D-20.

Table D-20. Duty Cycle MAC Sublayer Constants

Constant	Description	Default (symbols)	
		Channel Page 28 (channels 0-26) and Channel Page 29 (channels 27-34, 62) per EN300 220-1	Channel Page 30 (channels 35-61) and Channel Page 31 (channels 0-26) per EN300 220-1 & EN303 204
aDUTYCYCLEMeasurementPeriod	The period over which the duty cycle is calculated.	360,000,000 [3600s, 1hr]	
aDUTYCYCLEBuckets	Number of buckets used for duty cycle monitoring	13	
aDUTYCYCLERampUp	Time transmitter is transmitting carrier prior to start of data	Stack/silicon specific (in symbols)	
aDUTYCYCLERampDown	Time transmitter is transmitting carrier after end of data	Stack/silicon specific (in symbols)	

The MAC PIB Attributes required to support Duty Cycle Monitoring are given in Table D-21.

Table D-21. Duty Cycle MAC PIB Attributes

Attribute	Type	Range	Description	Default (symbols)
Read/Write Attributes				
mibDUTYCYCLELimited-Thresh	Integer	0 to *mibDUTYCY-CLERegulated*, but SHALL not exceed *mib-DUTYCYCLECritical-Thresh*	Threshold level, which if exceeded, identifies the limited duty cycle operation mode (in hundredths of a %).	5,400,000
mibDUTYCYCLECritical-Thresh	Integer	0 to *mibDUTYCY-CLERegulated*	Threshold level, which if exceeded, identifies the CRITICAL duty cycle operation mode.	7,500,000
mibDUTYCYCLERegu-lated	Integer	0 to 360,000,000 (3600s)	The regionally regu-lated maximum duty cycle permitted over the *aDUTYCY-CLEMeasure-mentPeriod*.	10,000,000 [100s, ~2.77%] for Channel Pages 28 & 29 9,000,000 [90s, ~2.5%] for Channel Pages 30 & 31
mibDUTYCYCLEUsed	Integer	0 to *mibDUTYCY-CLERegulated*	The current duty cycle used over the current measurement period *aDUTYCYCLEMeas-urementPeriod*.	0
mibDUTYCYCLEPtr	Integer	0 to *aDUTYCYCLEBuck-ets-1*	This is an index to the current bucket in the circular accumulator	0
mibDUTYCYCLEBucket[]	Integer ar-ray	0 - *mibDUTYCYCLEReg-ulated*	Array used as a circular accumulator of trans-mission time used in deriving transmission over past *aDUTYCY-CLEMeasure-mentPeriod*	0
mibDUTYCYCLEStatus	Enumera-tion	NORMAL, LIMITED CRITICAL, SUSPENDED	Current status of the duty cycle over the cur-rent *aDUTYCY-CLEMeasure-mentPeriod*.	NORMAL

D.13.1.2 Duty Cycle Calculations Over Measurement Period

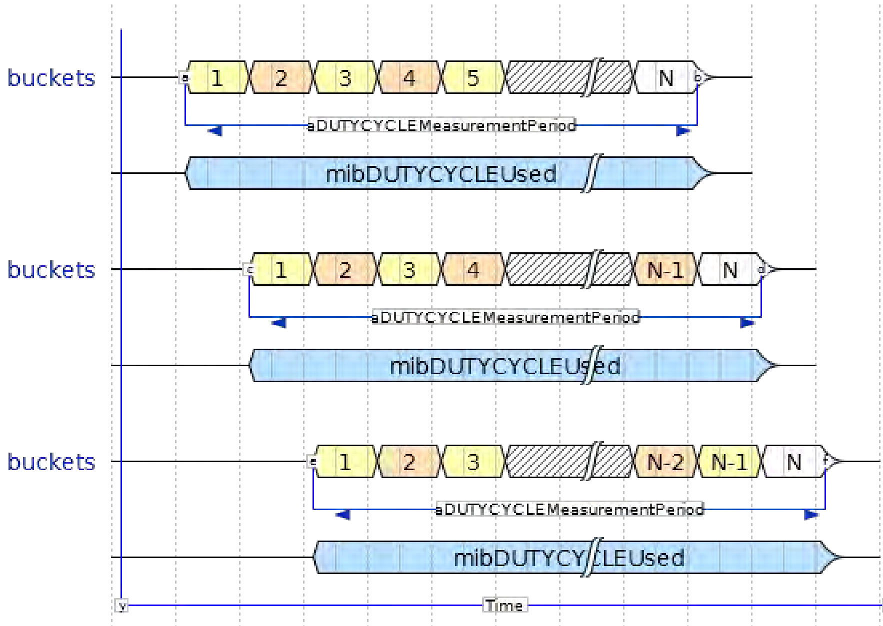
The duty cycle is calculated as the total transmitted time over the measurement period, i.e.:

$$\frac{mibDUTYCYCLEUsed}{aDUTYCYCLEMeasurementPeriod}\ \%$$

The duty cycle is with respect to the last *aDUTYCYCLEMeasurementPeriod* prior to the current time.

D.13.1.2.1 Accumulating the Duty Cycle Used

The duty cycle used is accumulated over the previous *aDUTYCYCLEMeasurementPeriod*, Accumulating over short periods and accumulating the sum over the previous N measurement periods the duty cycle is accurately and efficiently measured.



The number of accumulation buckets *aDUTYCYCLEBuckets*, that is chosen, has consequences on the accuracy and processing required. The following guidance is provided to:

- Reduce processing requirements
- Ensure it is not possible to exceed the duty cycle

In order to achieve these objectives it is suggested that the number of buckets is calculated by:

- Selecting a time period which is an integer factor of *aDUTYCYCLEMeasurementPeriod*
- Selecting *aDUTYCYCLEBuckets* such that:

$$aDUTYCYCLEBuckets = \left(\frac{aDUTYCYCLEMeasurementPeriod}{time\ period}\right) + 1$$

The consequences of the value *aDUTYCYCLEBuckets* are:

- The larger the number of time periods
    - o Higher resolution of duty cycle measurement
    - o Lower underutilized available transmission time available
    - o Increased processing
- The smaller the number of time periods
    - o Lower resolution of duty cycle measurement
    - o Higher underutilized available transmission time available
    - o Reduced processing

D.13.1.2.2 Examples of Selecting Values for *aDUTYCYCLEBuckets*

Measurement Periods			
aDUTYCYCLEBuckets	Resolution		Min Transmission Left (Seconds)
	in Seconds	in Minutes	
3	1800	30	0.0
13	300	5	0.0
25	150	2.5	0.0
61	60	1	40.0
241	15	0.25	85.0
3601	1	0.02	99.0

Assuming the oldest bucket is maxed out (lower of 100s or resolution) the table indicates the minimum time remaining of the 100s, until the measurement period is advanced at which point all the accumulated transmission in the oldest bucket will be freed.

There's a tradeoff to be made. Other algorithms can be used providing they can GUARANTEE that the maximum transmission in the measurement period is NEVER exceeded.

D.13.1.3 DC_CheckMode()

This routine updates the mode based on the accumulated duty cycle usage.

Parameters:

- NONE

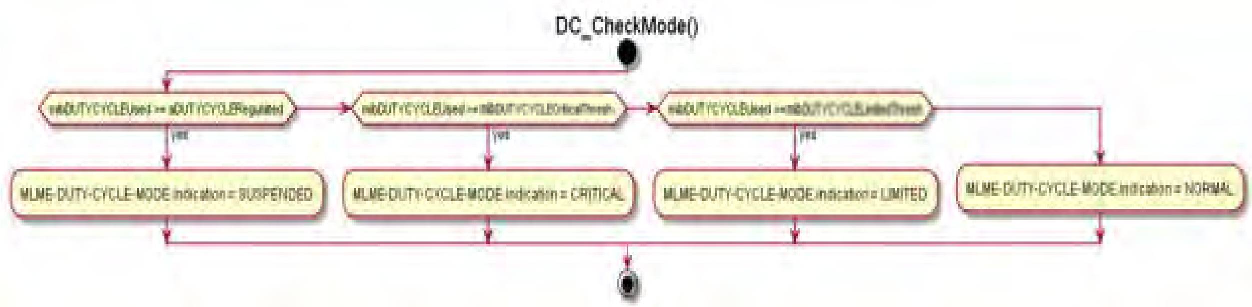
Returns

- NONE

Method:

1. Set MLME-DUTY-CYCLE-MODE.indication.

```
@startuml
title DC_CheckMode()
start
if (mibDUTYCYCLEUsed >= mibDUTYCYCLERegulated) then (yes)
: mibDUTYCYCLEStatus = SUSPENDED;
elseif (mibDUTYCYCLEUsed >= mibDUTYCYCLECriticalThresh) then (yes)
: mibDUTYCYCLEStatus = CRITICAL;
elseif (mibDUTYCYCLEUsed >= mibDUTYCYCLELimitedThresh) then (yes)
: mibDUTYCYCLEStatus = LIMITED;
else
: mibDUTYCYCLEStatus = NORMAL;
endif
stop
@enduml
```

D.13.1.4 DC_Bump Buckets

This routine handles the sliding integrator at the start of each time period used in monitoring the duty cycle usage.

Parameters:

- NONE

Returns

- NONE

Method:

1. Reduce the running total by the oldest bucket.
2. Clear the oldest bucket.
3. Update *mibDUTYCYCLEStatus* based on the running total.
4. If the *mibDUTYCYCLEStatus* attribute has changed send a MLME-DUTY-CYCLE-MODE.indication containing the new *mibDUTYCYCLEStatus* value.

D.13.1.5 Accumulate Transmit Time

This routine adds the transmission time to the duty cycle usage.

Parameters:

- symbolstransmitted - Number of symbols to add to the used transmission

Returns

- NONE

Method:

1. Add symbolstransmitted to the running total.
2. Add symbolstransmitted to the current accumulation bucket.
3. Update *mibDUTYCYCLEStatus* based on the running total.
4. If the *mibDUTYCYCLEStatus* attribute has changed send a MLME-DUTY-CYCLE-MODE.indication containing the new *mibDUTYCYCLEStatus* value.

D.13.1.6 Duty Cycle Monitoring Primitives Accessed Through the MLME-SAP

To support duty cycle monitoring, as required for GB Smart Energy and EU/UK Regional Sub-GHz FSK, the following primitives are required.

Name	**Request**	**Indication**	**Response**	**Confirm**
MLME-DUTY-CYCLE-MODE	-	D.13.1.6.1	-	-

D.13.1.6.1 MLME-DUTY-CYCLE-MODE.indication

The MLME-DUTY-CYCLE-MODE.indication primitive notifies the next higher level which duty cycle mode the device is currently operating in (NORMAL, LIMITED, CRITICAL, or SUSPENDED).

D.13.1.6.2 Semantics of the Service Primitive

The semantics of the MLME-DUTY-CYCLE-MODE.indication primitive are as follows:

```
MLME-DUTY-CYCLE-MODE          {
                              Status
                              }
```

Name	Type	Valid Range	Description
Status	Enumeration	NORMAL, LIMITED, CRITICAL, SUSPENDED	Is equal to the current value of *mibDUTYCYCLEStatus* and is used to indicate which duty cycle mode the device is currently operating in

D.13.1.6.3 When Generated

The MLME-DUTY-CYCLE-MODE.indication primitive indicates which duty cycle mode the device is currently operating in, (NORMAL, LIMITED, CRITICAL, or SUSPENDED) and is generated when the value of *mibDUTYCYCLEStatus* changes.

D.13.1.6.4 Duty Cycle Operating Modes

A MLME-DUTY-CYCLE-MODE.indication with the Status parameter set to NORMAL indicates that messages can be transmitted at the MAC layer and that the MAC queue is enabled and existing messages already in the MAC queue will be transmitted normally (subject to normal checking).

A MLME-DUTY-CYCLE-MODE.indication with the Status parameter set to LIMITED indicates that the Normal Operation Duty Cycle limit has been exceeded. Messages can be transmitted at the MAC layer and the MAC queue is enabled. Existing messages already in the MAC queue will be transmitted normally, however action MAY be taken, by the higher layers as a result of the notification by the MAC of entering the LIMITED mode, to reduce the duty cycle to prevent the interface from reaching the CRITICAL mode

A MLME-DUTY-CYCLE-MODE.indication with the Status parameter set to CRITICAL indicates that the Limited Operation Duty Cycle limit has been exceeded. Messages can be transmitted at the MAC layer and the MAC queue is enabled. Existing messages already in the MAC queue will be transmitted normally; however action MAY be taken, by the higher layers as a result of the notification by the MAC of entering the CRITICAL mode, to reduce the duty cycle to prevent the interface from reaching the SUSPENDED mode.

A MLME-DUTY-CYCLE-MODE.indication with the Status parameter set to SUSPENDED indicates that no more messages SHALL be transmitted at the MAC layer until an MLME-DUTY-CYCLE-MODE.indication is received with the Status parameter set to other than SUSPENDED. Each message in the MAC queue will be returned to the higher layer with a MCPS-DATA.confirm and a status of DUTY_CYCLE_EXCEEDED.

D.13.2 MAC Support for Listen Before Talk (LBT)

The MAC Sublayer Constants required to support LBT, due to regulatory limits, are given in Table D-22. The data rate for the GB Smart Energy and EU/UK Regional Sub-GHz FSK PHY is 100 kbps, giving a symbol period of 10μS. The values for the constants given below are given in both units of symbol periods and their corresponding time.

Table D-22. LBT MAC Sublayer Constants – Regulatory

Constant	Description	Channel Page 28 (channels 0-26) and	Channel Page 30 (channels 35-61)

		Channel Page 29 (channels 27-34, 62)	and Channel Page 31 (channels 0-26)
EN300-220 Regulatory Limits (units are symbols unless otherwise stated)			
aLBTTxMinOff	The minimum permitted off time between a device’s own transmissions	10,000 [100mS]	
aLBTTxMaxPKT	The maximum permitted transmission duration	100,000 [1S]	
aLBTMinFree	The minimum duration a channel SHOULD be free	500 [5mS]	16 [160µS]
aLBTMaxRandom	The maximum period of the backoff	500 [5mS]	500 [5mS]
aLBTMinRandom	The minimum period of the backoff	0	16 [160µS]
aLBTGranularity	The granularity in the random backoff	50 [500µS]	1 [10µS]
aLBTMaxDlg	The maximum Dialog period	400,000 [4S]	
aLBTThresholdLevelLp	The level (in dBm) at which the receiver determines there is activity in a low power channel (+14 dBm Tx).	-87 dBm	
aLBTThresholdLevelHp	The level (in dBm) at which the receiver determines there is activity in a high power channel (+27 dBm Tx).	-91 dBm	

Note: The UK have adopted the same values for ALL channel pages as defined for pages 28 & 29. Other regions MAY use different values for pages 30 & 31 which as yet have not been defined or verified.

By enabling LBT, as defined in EN 300-220 [B16], a device MAY increase its operational duty cycle from as low as 0.1% to ~2.5-2.7% (depending on the channel selected), which allows for support of increased traffic (messaging) from each device on a network.

EN 300-220 [B12] that, in order to transmit, a device SHALL WAIT until the channel is clear before monitoring the channel for at least *aLBTMinFree* and, if no traffic is seen, it MAY start transmitting. Otherwise it has to start the channel assessment again (with an additional random backoff).

In practice this means taking the following steps to determine if a device can transmit:

1. Wait until the channel becomes clear - the measured RSSI on the channel is below *aLBTThresholdLevelLp* or *aLBTThresholdLevelHp*, depending on whether the channel uses high power or low power devices (see).
2. Wait for *aLBTMinRandom* duration and then select a RANDOM time between zero and *aLBTMaxRandom*, with a granularity of *aLBTGranularity*.
3. Listen to channel for *aLBTMinFree* + RANDOM (as selected in step 2), if channel is not free in this period then go to step 1.
4. If the channel was free for the period in step 3 then start the transmission and return with status *LBTrcOk*, *otherwise* go to step 1.

Note(s):

1. If the channel does not become free within the time-out period *aLBTTimeout*, the transmission will not be started and it will return with error *LBTrcLBTBsy*.
2. During testing it has been decided to add a random backoff to the first attempt to access the channel to reduce the chance of collisions due to multiple devices and higher traffic than alternative implementations would see.

LBT is an attempt to provide reasonable access to the channel. In addition the *aLBTTxMinOff* hold off ensures no device can ever get 100% of the channel. A device may perform MAC retransmissions without waiting aLBTTxMinOff in between the retransmission, as long as the retransmissions do not exceed the aLBTTxMaxPKT duration.

The following sections describe an LBT mechanism which meets the requirements for the 863-870 MHz, 870-876 MHz & 915-921MHz bands as defined in EN 300-220 [B13] and EN 303-204 [B14].

D.13.2.1 LBT Specific MAC Constants and PIB Attributes - Implementation

The MAC Sublayer Constants required to support LBT implementation are given in Table D-23. The data rate for the GB SE and EU/UK Regional Sub-GHz FSK PHY is 100 kbps, giving a symbol period of 10µS. The values for the constants given below are given in both units of symbol periods and their corresponding time.

Table D-23. LBT MAC Sublayer Constants – Implementation

Constant*	**Description**	**Channel Page 28 (channels 0-26) and Channel Page 29 (channels 27-34, 62)**	**Channel Page 30 (channels 35-61) and Channel Page 31 (channels 0-26)**
Implementation (units are symbols unless otherwise stated)			
aLBTDlgResponseTimeout	The timeout if waiting for a reply during a Dialog sequence (SHALL be shorter than *aLBTMinFree*).	400 [4mS]	tbd
aLBTMaxTxRetries	The maximum number of retries allowed while looking for a clear channel. (The *aLBTTimeout* will probably mean 3 time retries will never be possible as each retry is 5-10mS, potentially 30mS + initial 5mS, i.e. up to 35mS.)	3	3
aLBTAckWindowStart	The minimum pause before acknowledging a received packet. This is to allow a transmitting device to change from transmit to receive mode. Starting an ACK before this time MAY result in the transmitter missing the ACK.	45 [450uS]	45 [450uS]
aLBTAckWindow	The maximum wait time before acknowledging a received packet (includes *aLBTAckWindowStart*). This time SHALL be shorter than aLBTMinFree otherwise other devices could interpret the quiet as an opportunity to transmit.	100 [1mS]	100 [1mS]
aLBTTimeout	Time before aborting LBT if it cannot find a free slot. [This value SHOULD be set to at least *aLBTMinFree* + *aLBTMaxTxRetries* * (*aLBTMinFree* + *aLBTMaxRandom*) + *aTxRxTurnAround* to ensure that all retries can occur.]	6000 [60mS]	6000 [60mS]
aRxTxTurnAround	Time for radio to switch between receive and transmit	100 [1000uS]	100 [1000uS]

Constant*	Description	Channel Page 28 (channels 0-26) and Channel Page 29 (channels 27-34, 62)	Channel Page 30 (channels 35-61) and Channel Page 31 (channels 0-26)
aTxRxTurnAround	Time for radio to switch between transmit and receive	45 [450uS]	45 [450uS]
aLBTTickMax	Ceiling at which Tick time base SHOULD be capped. This value SHALL be larger than *aLBTTxMinOff*	implementation decision	see description

NOTE: The UK have adopted the same values for ALL channel pages as defined for pages 28 & 29, Other regions MAY use different values for pages 30 & 31 which as yet have not been defined or verified.

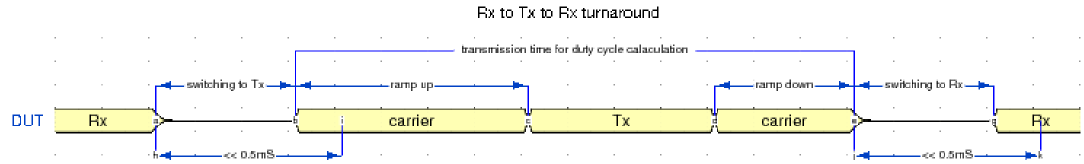
The MAC PIB Attributes required to support LBT implementation are given in Table D-24.

Table D-24. LBT MAC PIB Attributes - Implementation

Read/Write Attributes				
Attribute	**Type**	**Range (symbols)**	**Description**	**Default (symbols)**
mibLBTTxEndStamp	Integer	0-13,000 [130mS]	Time since the last transmission ceased.	0
mibLBTRxEndStamp	Integer	0-13,000 [130mS]	Time since the last transmission was received.	0
mibLBTTsStartDlg	Integer	0-1,400,000 [14S]	Time since the Dialog sequence started.	0
mibLBTDlgTimestartElapsed	Integer	0-400,000 [4S]	The elapsed time of the Dialog sequence.	0
mibLBTDialogMode	BOOLEAN	TRUE or FALSE	Set to TRUE when Dialog session is active	FALSE
mibLBTRadioState	Enum	OFF, Rx or Tx	Status of radio	OFF
mibLBTRadioWait	Integer	0 - 1000 [0-10mS]	Time left before a trans mission MAY occur	0
mibLBTTick	Integer	0 - 99999999	LBT tick timer	*0 or aLBTTickMax*

D.13.2.2 LBT Compliance when Changing Between Transmit/Receive Mode

In order for devices to operate within the LBT regulatory parameters and operate in a manner that will be interoperable and minimize collisions it is required that strict timing requirements are met when changing from one mode to another, The following diagram details these requirements:

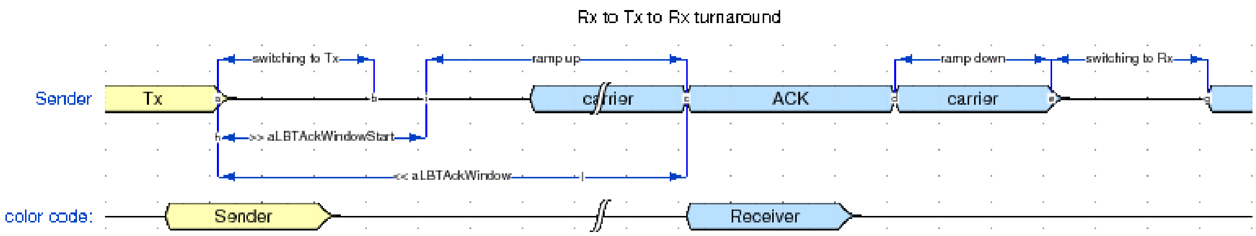


Rx to Tx - When the device is in listening mode and wishes to transmit it SHOULD switch from Rx to Tx mode and start transmitting either the carrier or the data portion such that other devices will detect the device is transmitting within *aRxTxTurnAround*, and SHALL account for propagation delays and like.

Tx to Rx - When the device is in transmit mode and switches to receive mode it SHOULD switch from Tx to Rx mode and start listening within *aTxRxTurnAround*, and SHALL account for propagation delays and like.

D.13.2.3 LBT and ACK Timing

In order to comply with LBT rules and minimize time and energy the ACK SHOULD be sent in accordance with the strict timing as shown in the following diagram:



If the receiver is to ACKnowledge the packet at the end of the transmission the receiver changes its radio to transmit and transmits the ACK. It is essential that the receiver does not:

- Attempt to transmit the ACK until at least *aLBTAckWindowStart* after receiving the request
- SHALL start the ACK transmission before *aLBTAckWindow*
- Return to Rx mode after transmitting the ACK

D.13.2.4 LBT Routines

The routines can be used for any of the LBT sub-bands and are used to illustrate how duty cycle and listen before talk (LBT) can be implemented.

The return/error codes for all LBT routines are given in Table D-25.

Table D-25. LBT Routines Return Codes

Error Code	#	Description of Code
LBTrcOk	0	Success
LBTrcLBTBsy	1	LBT timeout
LBTrcLBTMax	2	Maximum LBT retries limit reached
LBTrcTxMax	3	Maximum Tx retries limit reached
LBTrcTxTo	4	Tx timeout
LBTrcRxTo	5	Rx timeout
LBTrcAck	6	Valid ACK received

LBTrcAckTo	7	ACK transmission failed (missed the window to transmit)
LBTrcRxInv	8	Invalid message received
LBTrcTxToPkt	99	Single transmission exceeded
LBTrcTxToMax	100	Transmissions have exceeded the maximum rate in the last hour

The following are the main and auxiliary routines required to support LBT.

The **LBT_ACK** and **LBT_TX** routines describe how the transmit time, due to any type of transmission (ACK, MAC, etc.), gets accumulated in *mibDUTYCYCLEUsed*. They do not however describe how transmit time more than an hour old gets decremented from *mibDUTYCYCLEUsed*.

D.13.2.4.1 LBT_Tick()

This routine is used to read the LBT timebase whose resolution is symbols (ie 10uS units).

Parameters:

- NONE

Returns

- Current value of *mibLBTTick*

Method:

1. Return contents of *mibLBTTick.*

D.13.2.4.2 LBT_TickReset()

This routine resets the LBT timebase.

Parameters:

- NONE

Returns

- NONE

Method:

1. Set *mibLBTTick* = 0.

D.13.2.4.3 LBT_TickIncrement()

This routine increments the LBT timebase and is typically invoked by an interrupt whose period is either a symbol or a multiple eg 8 for a byte (using higher increments can reduce interrupt overhead).

Parameters:

- incr – increment

Returns

- NONE

Method:

1. *mibLBTTick* += incr

D.13.2.4.4 LBT_TickExhaust()

This routine expires the LBT timebase and is typically called during the initialization of the stack. This ensures that the device does not have to wait (other than the aLBTMinOff) for the first transmission. Sleepy devices that have not transmitted for a significant time can also call this to ensure they are able to transmit asap.

Parameters:

- NONE

Returns

- NONE

Method:

1. Set *mibLBTTick* = *aLBTTickMax*

D.13.2.4.5 LBT_Backoff()

This routine addresses Requirements - EN 300 220-1 :: [9.2.2]

Parameters:

- NONE

Returns:

- A random time period in symbols

Method:

1. Generate a random period from *aLBTMinRandom* to *aLBTMaxRandom*, with a granularity of *aLBTGranularity*.

D.13.2.4.6 LBT_ACK()

This routine addresses Requirements - EN 300 220-1 :: [3.1], [7.10], [9.2], [9.2.4], [9.2.5], [9.2.5.2.3]

Parameters:

- NONE

Returns:

- Status

Local variables:

- timeout - used in determining if ACK fails to be transmitted before the end of the ACK window
- aPPDUs - number of symbols required to transmit ACK

Method:

To transmit an ACK it is necessary to:

1. Set timeout = **LBT_Tick()** + *aLBTAckWindow*.
2. Call **LBT_RadioTx(***aLBTAckWindowStart***)** to Switch radio to transmit and delay to start of ACK window.
3. Calculate the number of symbols required to be transmitted, i.e. set aPPDUs = 8*packet size (bytes) + *aDUTYCYCLERampUp* + *aDUTYCYCLERampDown*.
4. Check the total hourly transmission time will not be exceeded, i.e. (*mibDUTYCYCLEUsed* plus aPPDUs) is less than *aLBTTxMaxPeriod*.
   a. If it would exceed the hourly limit, then switch radio to Rx, and return error *LBTrcTxToMax*.
5. Check that the transmitter will not be transmitting beyond the maximum time allowed for a single transmission, i.e. aPPDUs is less than *aLBTTxMaxPKT*.
   a. If it would exceed the time allowed for a single transmission, then switch radio to Rx, i.e. call **LBT_RadioRx()** and return error *LBTrcTxToPkt*.
6. While (! **LBT_RadioReady()**){/*wait*/}//Wait for radio to be ready to transmit.
7. If *aLBTAckWindow* time has elapsed since the last bit was received, i.e. if (*mibLBTRxEndStamp* + *aLBTAckWindow*) is less than **LBT_Tick()**:
   a. Change radio to receive.
   b. Return with error code *LBTrcAckTo*.
8. Transmit the ACK PPDU - normal 802.15.4 raw transmit code.
9. Change radio to receive.
10. Increment *mibDUTYCYCLEUsed* by aPPDUs.
11. Return *LBTrcOk*.

D.13.2.4.7 LBT_LBT()

This routine addresses Requirements - EN 300 220-1 :: [7.10], [8.2], [9.1], [9.2], [9.2.2]

Parameters:

- timeout

Returns:

- Error Code or Success Code

Local variables:

- timeout

Method:

To determine if a transmission can be initiated it is necessary to:

1. Set timeout to **LBT_Tick()** + timeout.
2. If the radio is not in Rx mode then set radio to Rx mode.

3. Wait until the last transmission was at least *aLBTTxMinOff* earlier, i.e. *mibLBTTxEndStamp* plus *aLBTTxMinOff* minus *aLBTMinFree is less than or equal to current tick time* **LBT_Tick()**.
4. Set count to 0.
5. Wait for radio to be ready to receive.
6. While (count ≤ *aLBTMaxTxRetries AND timeouttimer < aLBTTimeout)*:
    a. Check if channel is busy for (*aLBTMinFree* plus **LBT_Backoff()**) symbols.
    b. If channel has been quiet for the entire period, then turn on the transmitter and exit, i.e. call LBT_RadioTx(0) and return with status *LBTrcOk*.
    c. Increment count by 1.
    d. Go back to Step 6.
7. If *aLBTMaxTxRetries* is exceeded return status *LBTrcLBTMax*.
8. Return status *LBTrcLBTBsy*.

D.13.2.4.8 LBT_TX() (to be included as part of regular TX operation)

This routine addresses Requirements - EN 300 220-1 :: [3.1], [7.10], [8.2], [9.1], [9.2], [9.2.2], [9.2.5]

Parameters:

- Message to be transmitted, PPDU
- TxTimeOut – time period in which the transmitter SHOULD have been able to transmit a message

Returns:

- Error Code or Success Code

Local variables:

- count - error counter
- dPPDUs - number of symbols required to transmit data

Method:

To transmit a data packet (PPDU) it is necessary to obey the LBT requirements, i.e.:

1. If the radio is not on then switch it on in receive mode.
2. Calculate the number of symbols required to transmit: i.e. dPPDUs = 8*packet size (bytes) + *aDUTYCYCLERampUp* + *aDUTYCYCLERampDown*.
3. Ensure the total hourly transmission time will not be exceeded, i.e. (*mibDUTYCYCLEUsed*+dPPDUs less than *aLBTTxMaxPeriod)*.
    a. If it would, then return error *LBTrcTxToMax*.
4. Check that the transmitter will not be transmitting beyond the maximum time allowed for a single transmission, i.e. dPPDUs less than *aLBTTxMaxPKT*:
    a. If it would, then return error *LBTrcTxToPkt*.

    [When/if DIALOG mode is defined Check DIALOG .
5. Determine if channel is clear.
    a. Perform LBT assessment as described in **LBT_LBT()**.
    b. If channel is quiet, then exit returning *LBTrcBsy*.
6. If TxTimeOut exceeded, return *LBTrcTxTo*.

7. Recheck that the total hourly transmission time will not be exceeded, i.e. (*mibDUTYCYCLEUsed*+dPPDUs less than *aLBTTxMaxPeriod)*.
    a. If it would, then switch radio back to Rx and return error *LBTrcTxToMax*.
8. Wait for radio to stabilize.
9. Transmit the data packet PPDU - normal 802.15.4 raw transmit code.
10. Turn off the transmitter and enable the receiver.
11. Reset the counter for *mibLBTTxEndStamp* to zero.
12. If not in Dialog mode reset the tick counter. All timing is relative to the start of transmission after LBT unless in Dialog mode, in which case it is relative to the first transmission in the Dialog sequence.
13. Add dPPDUs to *mibDUTYCYCLEUsed*.
14. Wait to see if an ACK is received (SHOULD start receiving ACK before *aLBTAckWindow)*.
15. If an ACK is returned then return *LBTrcOk*.

D.13.2.4.9 LBT_RX()

This routine addresses Requirements - EN 300 220-1 :: [9.2.4], [9.2.5]

Parameters:

- RxTimeOut – time period after which the receiver SHOULD abort if no message is being received

Returns:

- Status Code
- A PPDU message if successful

Local variables:

- timeout

Method:

Listens for valid messages and then returns either due to a RxTimeOut, valid ACK or message received.

1. Ensure radio is in receive mode, i.e. call **LBT_RadioRx()**.
2. Listen for message – normal 802.15.4 raw receive code.
    a. If RxTimeOut exceeded and no message being received then return with *LBTrcRxTo*.
3. Message received, reset the counter for *mibLBTRxEndStamp* to **LBT_Tick()**.
4. Verify message, (crc, addressed to me or unicast or multicast).
5. If message is a valid ACK them return *LBTrcAck*.
6. If message is invalid and time still left to receive another message go to Step 2.
    a. Otherwise return *LBTrcRxInv*.
7. If packet SHOULD be acknowledged Call **LBT_ACK()** and return its return code, otherwise return *LBTrcOk*.

D.13.2.4.10 LBT_RadioRx()

This routine puts the radio in Receive mode. If the radio is off it's necessary to delay use till it switches on, if it needs to change from Tx to Rx then it will have to settle changing mode, additionally an alternative delay could be requested and this will be used if its larger than delay due to state change.

Parameters:

- delay - alternative waiting time - pass 0 for no alternative delay

Returns:

- NONE

Method:

1. If radio is off:
    a. Turn radio on.
    b. Set *mibLBTRadioWait* to **LBT_Tick()**.
    c. Add largest of *aRxOnTime* or delay to *mibLBTRadioWait.*
2. Else if *mibLBTRadioState* is Tx:
    a. Switch radio to Rx.
    b. Set *mibLBTRadioWait* to **LBT_Tick()**.
    c. Add largest of *aTxRxTurnAround* or delay to *mibLBTRadioWait*.
3. Else if *mibLBTRadioState* is Rx:
    a. If delay > 0 then set *mibLBTRadioWait* to **LBT_Tick()**+delay.
4. Set *mibLBTRadioState* to Rx.

D.13.2.4.11 LBT_RadioTx()

This routine puts the radio in Transmit mode. If the radio is off its necessary to delay use till it switches on, if it needs to change from Rx to Tx then it will have to settle changing mode, additionally an alternative delay could be requested and this will be used if its larger than delay due to state change.

Parameters:

- delay - Alternative waiting time, e.g. LBT ACK window start, pass 0 for no alternative delay

Returns:

- NONE

Method:

1. If radio is off:
    a. Turn radio on and to Tx.
    b. Set *mibLBTRadioWait* to **LBT_TICK()**.
    c. Add largest of *aRxOnTime* or delay to *mibLBTRadioWait.*
2. Else if *mibLBTRadioState* is Rx:
    a. Switch radio to Tx.
    b. Set *mibLBTRadioWait* to **LBT_TICK()**.
    c. Add largest of *aRxTxTurnAround* or delay to *mibLBTRadioWait.*
3. Else if *mibLBTRadioState* is Tx:
    a. If delay > 0 then set *mibLBTRadioWait* to **LBT_TICK()** + delay.
4. Set *mibLBTRadioState* to Tx.

D.13.2.4.12 LBT_RadioReady()

This routine checks the radio is ready after being powered on or/changed mode.

Parameters:

- NONE

Returns:

- 0 - not ready
- 1 - Radio is ready to use

Method:

1. If *mibLBTRadioWait* is greater than or equal to **LBT_TICK()** then set *mibLBTRadioWait* to 0.
2. If *mibLBTRadioWait* is equal to 0 then return 1.
3. Return 0.

D.13.2.4.13 LBT_RadioOff()

This routine turns the radio off.

Parameters:

- NONE

Returns:

- NONE

Method:

1. If radio is on then turn it off.
2. Set *mibLBTRadioState* to OFF.

D.13.2.5 Consequence of LBT on Devices

LBT and the *aLBTTxMinOff* have many side effects. The following describes some of the potential scenarios:

Scenario1 - Normal sequence assuming non Dialog mode

1. The initiator sends request after performing LBT on channel.
2. The responder MAC ACK's the request.
3. Depending on when the responder last transmitted it will not be able to respond until *aLBTMinFree* to *aLBTMinOff* symbols.
4. The responder waits for a clear channel and then transmits to the initiator.
5. The initiator immediately responds with a MAC ACK.
6. If the initiator wishes to send another request it depends on when the initiator last transmitted as to when it will be able to send another request. The time in which the initiator will be able to send another request is somewhere between *aLBTMinFree* and *aLBTMinOff* symbols.
7. If the initiator is sending another message go to step 1.

D.13.2.6 Notes on LBT

On reception of a packet a device can immediately ACK the packet only delaying long enough to allow the originating device to switch to receive mode (ie *aLBTAckWindowStart*).

D.14 Regional Sub-GHz FSK PHY Specification

D.14.1 Regional Sub-GHz PIB Attributes

In IEEE Std 802.15.4-2020 the SUN PHY specifications define physical layer parameters within the information fields of the SUN Device Capabilities IE. In order to maximize use of the IEEE Std 802.15.4 specification the following information fields have been mapped to PIB attributes with like names in Table D-26. PIB attribute types are also defined in Table D-26. The Regional Sub-GHz as defined here deviates from 15.4 in that it uses the RS-GFSK MCS mode 4 500 kbps RS-GFSK PHY for N.A. in lieu of the 100 kbps PHY defined by the SUN PHY specifications. This is done in order to allow for a +30 dBm Tx power without requiring frequency hopping.

Table D-26. Information Field to Regional Band PHY PIB Attributes Mapping

SUN PHY Information Field	Regional Band PIB Attribute	Type	Range
Enh-ACK	*phyEnhAck*	Integer	0,1
Data Whitening	*phyDataWhitening*	Integer	0,1
Interleaving	*phyInterleaving*	Integer	0,1
SFD G1	*phySfdG1*	Integer	0,1
NRNSC FEC	*phyNrnscFec*	Integer	0,1
RSC FEC	*phyRscFec*	Integer	0,1
Mode Switch	*phyModeSwitch*	Integer	0,1
Extended Band Identifier	*phyExtendedBandIdentifier*	Integer	0,1
Frequency Bands Supported	*phyFrequencyBandsSupported*	Bitmap	4 octet
PHY Type	*phyPhyType*	Integer	0-15
All Frequency Bands	*phyAllFrequencyBands*	Integer	0,1
PHY Modes Supported	*phyPhyModesSupported*	Bitmap	11 bits
Specific Frequency Bands	*phySpecificFrequencyBands*	Bitmap	4 octet

Descriptions for each of the Regional Band PHY PIB Attributes are the same as the SUN PHY Information Field from which they are being mapped, unless defined differently below (those preceded by an *).

phyEnhAck SHALL be set to 1 (signifying that Enh-ACK frames are supported).

phyDataWhitening SHALL be set to 1 (signifying data whitening of the PSDU is always enabled).

phyInterleaving SHALL be set to 0 (signifying interleaving is not supported).

phySfdG1 SHALL be set to 0 (signifying Group 0 SFD is supported).

phyNrnscFec SHALL be set to 0 (signifying NRNSC FEC is not supported).

phyRscFec SHALL be set to 0 (signifying RSC FEC is not supported).

phyModeSwitch SHALL be set to 0 (signifying that only a single data rate is used and that mode switching is not supported).

phyExtendedBandIdentifier SHALL be set to 1 (signifying extended frequency band identifier values are supported).

phyFrequencyBandsSupported SHALL be set to all possible Frequency Band Identifiers the device is capable of supporting.

phyPhyType SHALL be set to 1 (signifying use of modulation scheme FSK-B).

**phyAllFrequencyBands* SHALL be set to 0 (signifying the PHY Type is only supported in the frequency band declared in the Specific Frequency Bands field).

phyPhyModesSupported SHALL be set to 0x080 (signifying PHY Mode 7, for 500 kb/s; 2-FSK; mod index = 0.76; channel spacing = 1 MHz in the N.A. Region, and for 100 kb/s; 2-FSK; mod index = 0.5; channel spacing = 200 kHz in Euro Region).

**phySpecificFrequencyBands* is a bitmap that SHALL be set to indicate the specific Frequency Band Identifiers listed in *phyFrequencyBandsSupported* in which the device is currently operating in.

The information fields in Table D-26 SHALL take the same value as their mapped PIB attributes.

D.14.2 Regional Sub-GHz FSK Frame Format

The reader is referred to section D.12.1 for the frame format that SHALL be used for Regional Sub-GHz FSK.

D.14.3 Regional Sub-GHz FSK PHY

The Regional Sub-GHz PHY's define sub-GHz PHYs for N.A. and European band use which provide a higher data rate than the PHY's defined by IEEE Std 802.15.4-2003, which are aligned with IEEE Std 802.15.4-2020. The Euro regional sub-GHz PHY is currently only specified by Zigbee for use in Europe.

D.14.3.1 Modulation Specification

N.A. Regional Sub-GHz FSK SHALL use the modulation requirements specified in Table D-27.

Table D-27. N.A. Regional Sub-GHz Modulation Requirements

Parameter	Configuration
Modulation	2-Level GFSK
Data Rate	500 kbps
Tx Filter BT	0.5 (Gaussian)
Modulation Index	0.76
Channel Spacing	1.0 MHz

Operation in the N.A. 902-928 MHz band is regulated by FCC Part 15.247 rules. Typically a modulation index of 0.5 is used, unless there are other considerations. However, when this PHY was being developed in 802.15 the task group wanted to make sure that when including tolerances, drifts, etc. that the modulation would always meet the minimum 500 kHz BW as specified by the FCC, therby allowing single channel operation and not having to follow the hopping rule. Analysis/consideration of all factors led the task group to choose a Modulation Index 0.76.

Euro Regional Sub-GHz FSK SHALL use the modulation requirements specified in Table D-28.

Table D-28. Euro Regional Sub-GHz Modulation Requirements

Parameter	Configuration
Modulation	2-Level GFSK
Data Rate	100 kbps
Tx Filter BT	0.5 (Gaussian)
Modulation Index	0.5
Channel Spacing	200 kHz

No other modulation parameters are supported.

D.14.3.2 Forward Error Correction (FEC)

Regional Sub-GHz FSK SHALL NOT use FEC coding.

D.14.3.3 Data Whitening

Regional Sub-GHz FSK devices SHALL support Data whitening as specified in 20.4 of IEEE Std 802.15.4-2018.

D.14.3.4 Channels and Frequencies

Regional Sub-GHz FSK devices SHALL be capable of operating on all the channels specified, in all of the bands specified in Table D-29.

Table D-29. Regional Sub-GHz FSK PHY Band Parameters

Frequency Band Identifier*	**Frequency Band (MHz)**	**Region**	**ChanSpac-ing (MHz)**	**TotalNum-Chan**	**Channel #'s Used**	**1st ChanCenterFreq (MHz)**
16	902-928	North America and Mexico	1.0	25	0-24	903.0
19	915-921	Europe	0.2	21	56-76	915.2
4	863-870	Europe	0.2	35	0-34	863.1
15	870-876	Europe	0.2	21	35-55	870.2

*The *FrequencyBandIdentifier* is defined by IEEE 802.15

The exact Channel Plan used in a Regional Sub-GHz FSK deployment SHALL be specified using a channel mask.

D.14.3.4.1 Channel Numbering

D.14.3.4.1.1 NA Regional Sub-GHz FSK PHY Channels

Channel center frequencies corresponding to the channel numbers for the N.A. Regional Sub-GHz FSK PHY are specified as follows:

For the 902 MHz - 928 MHz band:

$$ChanCenterFreq = ChanCenterFreq0 + NumChan * ChanSpacing$$

where

$$ChanCenterFreq0 = 903.0 \text{ MHz}$$

where ChanCenterFreq0 is the *first* channel center frequencies in MHz, *ChanSpacing* is the separation between adjacent channels in MHz, and *NumChan* is the channel number.

For the 902-928 MHz Band defined in Table D-29:

- *ChanCenterFreq0* (MHz) = 903.0
- *ChanSpacing* (MHz) = 1.0
- *TotalNumChan* = 25
- *NumChan* (Chan. #) goes from 0 to 24

This results in the following channel numbers and center frequencies for the 902-928 MHz Band shown in Table D-30, which SHALL be used for the 902-928 MHz Band:

Table D-30. Channels and Center Frequencies for 902-928 MHz Band Designation

Channel #	Fc (MHz)
0	903.0
1	904.0
…	…
23	926.0
24	927.0

D.14.3.4.1.2 Euro Regional Sub-GHz PHY Channels

Channel center frequencies corresponding to the channel numbers specified for the Euro Regional Sub-GHz FSK PHY are specified as follows:

For the 863 MHz - 870 MHz band

$$\mathit{ChanCenterFreq} = \mathit{ChanCenterFreq0} + (\mathit{NumChan} * \mathit{ChanSpacing})$$

where

$$\mathit{ChanCenterFreq0} = 863.1\ \mathit{MHz}$$

For the 870 MHz - 876 MHz band

$$\mathit{ChanCenterFreq} = \mathit{ChanCenterFreq35} + (\mathit{NumChan} - 35) * \mathit{ChanSpacing}$$

where

$$\mathit{ChanCenterFreq35} = 870.2\ \mathit{MHz}$$

For the 915 MHz - 921 MHz band

$$\mathit{ChanCenterFreq} = \mathit{ChanCenterFreq56} + (\mathit{NumChan} - 56) * \mathit{ChanSpacing}$$

where

$$\mathit{ChanCenterFreq56} = 915.2\ \mathit{MHz}$$

where *ChanCenterFreq0*, *ChanCenterFreq35* and *ChanCenterFreq56 are* the first channel center frequencies in MHz, *ChanSpacing* is the separation between adjacent channels in MHz, and *NumChan* is the channel number.

For the 863-870 MHz Band defined in Table D-29:

- *ChanCenterFreq0* (MHz) = 863.1
- *ChanSpacing* (MHz) = 0.2
- *TotalNumChan* = 35
- *NumChan* (Chan. #) goes from 0 to 34

This results in the following channel numbers and center frequencies for the 863-870 MHz Band shown in Table D-31, which SHALL be used for the 863-870 MHz Band.

Table D-31. Channels and Center Frequencies for 863-870 MHz Band Designation

Channel #	Fc (MHz)
0	863.1
1	863.3
…	…
33	869.7
34	869.9

For the 870-876 MHz Band defined in Table D-32:

- *ChanCenterFreq35* (MHz) = 870.2
- *ChanSpacing* (MHz) = 0.2
- *TotalNumChan* = 21
- *NumChan* (Chan. #) goes from 35 to 55

This results in the following channel numbers and center frequencies for the 870-876 MHz Band shown in Table D-32, which SHALL be used for the 870-876 MHz Band Designation:

Table D-32. Channels and Center Frequencies for 870-876 MHz Band Designation

Channel #	Fc (MHz)
35	870.2
36	870.4
…	…
54	874.0
55	874.2

For the 915-921 MHz Band defined in

- *ChanCenterFreq56* (MHz) = 915.2
- *ChanSpacing* (MHz) = 0.2
- *TotalNumChan* = 21
- *NumChan* (Chan. #) goes from 56 to 76

This results in the following channel numbers and center frequencies for the 915-921 MHz Band shown in Table D-33, which SHALL be used for the 915-921 MHz Band.

Table D-33. Channels and Center Frequencies for 915-921 MHz Band Designation

Channel #	Fc (MHz)
56	915.2
57	915.4
…	…

Channel #	Fc (MHz)
75	919.0
76	919.2

D.14.3.4.2 Channel Pages

Six channel pages are allocated for Regional Sub-GHz FSK, using the existing 32-bit mechanism (5-bits of channel-page, 27-bits of channel-mask), spreading the channels across the 6 pages as follows.

Channel Page	Description	Channels #'s Used
23	902-928 MHz band, channels 0-24	0-24
24	915-921 MHz band, channels 56-76	56-76
25	863-870 MHz band, channels 0-26	0-26
26	863-870 MHz band, channels 27-34	27-34
27	870-876 MHz band, channels 35-55	35-55

D.14.3.5 Regional Sub-GHz FSK RF Requirements

D.14.3.5.1 Receiver Requirements

D.14.3.5.1.1 Standard Measurement Conditions

The Standard Measurement Conditions for all receiver requirements SHALL be:

- 139 byte packets (8 bytes Preamble + 2 byte SFD + 2 byte PHR + 127 byte PSDU)
- Receiver power measurements made at the antenna connector
- Packet Error Rate of less than 1%

The PHY RF requirements, when measured under these conditions, are intended to apply to a typical device rather than the worst sample of a batch.

D.14.3.5.1.2 Sensitivity Requirement

Under the Standard Measurement Conditions, Euro Regional Sub-GHz FSK SHALL meet a Reference Sensitivity as specified in Table D-34.

Table D-34. Euro Regional Sub-GHz Receiver Reference Sensitivity Requirement

Reference Sensitivity (dBm)
-99

Under the Standard Measurement Conditions, N.A. Regional Sub-GHz FSK SHALL meet a Reference Sensitivity as specified in Table D-35.

Table D-35. N.A. Regional Sub-GHz Receiver Reference Sensitivity Requirement

Reference Sensitivity (dBm)
-91

D.14.3.5.1.3 Co-Channel Rejection Requirement

Under the Standard Measurement Conditions, with the wanted signal at 20 dB above the respective regions Reference Sensitivity level and with an interfering signal modulated with the same modulation as the wanted signal and on the same channel, Regional Sub-GHz FSK receivers SHALL meet their respective regions Reference Sensitivity PSR with the co-channel rejection requirement for the interferers level relative to the wanted signal level as specified in Table D-36.

Table D-36. Receiver Co-channel Rejection Requirement

Co-channel Rejection (dB)
-15

D.14.3.5.1.4 Selectivity Requirements

Under the Standard Measurement Conditions, with the wanted signal at 3 dB above the respective regions Reference Sensitivity level and with an interfering signal modulated with the same modulation as the wanted signal and at the frequency offsets specified in below, Regional Sub-GHz FSK receivers SHALL meet their respective regions Reference Sensitivity PSR with the selectivity requirement for the interferers level relative to the wanted signal level as specified in Table D-37.

Table D-37. Receiver Selectivity Requirements

Channel Offset	Level of Interferer relative to wanted (dB)
+/- 1 Channel	5
+/- 2 Channel	30

D.14.3.5.2 Receive Power Level (RSSI)

The receiver SHALL be capable of measuring the received power level (reported as the RSSI) of a packet (measured over any portion of the received packet), on a packet by packet basis.

Euro Regional Sub-GHz FSK operations SHALL be capable of measuring the received power over at least the range defined in Table D-38.

Table D-38. Non N.A. Regions Receive Power Measurement Range

Receive Power Measurement Lower Minimum	Receive Power Measurement Upper Minimum
-97 dBm	-50 dBm

N.A. Regional Sub-GHz FSK operations SHALL be capable of measuring the received power over at least the range defined in Table D-39.

Table D-39. N.A. Region Receive Power Measurement Range

Receive Power Measurement Lower Minimum	Receive Power Measurement Upper Minimum
-89 dBm	-42 dBm

The receiver SHALL be capable of measuring the received power with the step size and accuracy defined in Table D-40.

Table D-40. Receive Power Measurement Step Size and Accuracy

Receive Power Measurement Step Size	Receive Power MeasurementAccuracy
≤ 2 dB	≤ 3 dB

D.14.3.5.3 Transmitter Requirements

D.14.3.5.3.1 Maximum Transmit Power

The Regulated Maximum Allowable Transmit Power for European Regional Sub-GHz FSK devices at the transmitter SHALL be as specified below in Table D-41.

Table D-41. Maximum Transmit Power for Euro Regional Sub-GHz

Maximum Transmit Power
+14 dBm

The Regulated Maximum Allowable Transmit Power for NA Sub-GHz FSK devices at the transmitter SHALL be capable of transmitting the Maximum Transmit Power as specified below in Table D-42.

Table D-42. Maximum Transmit Power for N.A. Region

Maximum Transmit Power
+30 dBm

D.14.3.5.3.2 Minimum Transmit Power

The Minimum Transmit Power for Regional Sub-GHz FSK operation, for all the channels, SHALL be as specified in Table D-43.

Table D-43. Minimum Transmit Power

Minimum Transmit Power
-15 dBm

D.14.3.5.3.3 Transmit Power Step Size and Accuracy

The transmitter SHALL be capable of adjusting its transmit power over the range from the Minimum Transmit Power to the maximum power of the device or the maximum specified power, whichever is reached first, with the step size and accuracy specified in Table D-44.

Table D-44. Transmit Power Step Size and Accuracy

Transmit PowerStep Size	Transmit PowerAccuracy
≤ 2 dB	≤ 3 dB

D.14.3.5.4 Link Quality Assessment (LQA)

The raw LQA value is calculated per incoming packet, as follows:

$$\mathrm{LQA}_{raw}(c,r)=255\times\frac{c-c_{min}}{c_{max}-c_{min}}\times\frac{r-r_{min}}{r_{max}-r_{min}}$$

Here, c is the cross-correlation peak of the PHY preamble sequence, or the LQI value reported by the IEEE Std 802.15.4-2015 MAC and PHY, or another suitable signal quality indicator in the range $[c_{min},c_{max}]$, and r is the received signal strength indicator (RSSI) in the range $[r_{min},r_{max}]$. The resulting raw LQA value in the range

[0,255], where 0 denotes a critical link, 255 denotes an excellent link, incorporates both the received signal strength and the signal quality to assess the longer-term quality and stability of the link by taking receiver capabilities into account. This is meant to flatten the step-like MAC layer LQI function, which typically yields high values over 95% of the link-budget and drastically drops beyond this point. In contrast, LQA is designed to provide a linear quality estimate over the entire link-budget, so it can be used to better rank links to different peers. Specifically, if the received signal strength is close to receiver sensitivity but signal quality is still high (clear signal), LQA in contrast to MAC layer LQI would classify the link as critical.

These bounds SHALL be determined on final production hardware as follows:

- c_min is the lowest signal quality ever reported, i.e. for a packet that can barely be received,
- c_max is the highest signal quality ever reported, i.e. for a packet received under ideal conditions,
- r_min is the lowest signal strength ever reported, i.e. for a packet close to receiver sensitivity,
- r_max is the highest signal strength ever reported, i.e. for a packet received from a strong, close-by transmitter.

Values of c and r, as determined at runtime during normal operation of the device, SHALL be bound by their respective minimum and maximum values prior to feeding them into the 〚LQA〛_raw (c,r) formula above.

For links, which belong to entries in the neighbor table, implementations SHOULD apply filtering of per-packet LQA values as specified in section 3.6.4.3 in order to produce final LQA values prior to subsequent processing, e.g. routing cost calculations.

ANNEX E OPERATING NETWORK MANAGER AS NETWORK CHANNEL MANAGER FOR INTERFERENCE REPORTING AND RESOLUTION

Prerequisites: Devices SHALL limit their operations to channels within their current PHY (i.e. 868/915 MHz or 2450 MHz). Commands including channels outside the band shall be ignored.

A single device can become the Network Channel Manager. This device acts as the central mechanism for reception of network interference reports and changing the channel of the network if interference is detected. The default address of the network manager is the coordinator, however this can be updated by sending a Mgmt_NWK_Update_req command with a different short address for the network channel manager. The device that is the Network Channel Manager SHALL set the network manager bit in the server mask in the node descriptor and SHALL respond to System_Server_Discovery_req commands.

Each router or coordinator is responsible for tracking transmit failures using the TransmitFailure field in the neighbor table and also keeping a NIB counter for total transmissions attempted. A device that detects a significant number of transmission failures MAY take action to determine if interference is a cause. The following steps are an example of that procedure:

1. Conduct an energy scan on all channels within the current PHY. If this energy scan does not indicate higher energy on the current channel then other channels, no action is taken. The device SHOULD continue to operate as normal and the message counters are not reset. However, repeated energy scans are not desirable as the device is off the network during these scans and therefore implementations SHOULD limit how often a device with failures conducts energy scans.
2. If the energy scan does indicate increased energy on the channel in use, a Mgmt_NWK_Unsolicited_Enhanced_Update_notify SHOULD be sent to the Network Manager to indicate interference is present. This report is sent as an APS Unicast with acknowledgement and once the acknowledgment is received the total transmit and transmit failure counters are reset to zero.
3. To avoid a device with communication problems from constantly sending reports to the network manager, the device SHOULD NOT send a Mgmt_NWK_Unsolicited_Enhanced_Update_notify more than 4 times per hour.

Upon receipt of a Mgmt_NWK_Unsolicited_Enhanced_Update_notify message, the network manager SHALL evaluate if a channel change is required in the network. The specific mechanisms the network manager uses to decide upon a channel change are left to the implementers. It is EXPECTED that implementers will apply different methods to best determine when a channel change is required and how to select the most appropriate channel. The following is offered as guidance for implementation.

1. The network manager MAY do the following:
2. Wait and evaluate if other reports from other devices are received. This MAY be appropriate if there are no other failures reported. In this case the network manager SHOULD add the reporting device to a list of devices that have reported interference. The number of devices on such a list would depend on the size of the network. The network manager can age devices out of this list.
3. Request other interference reports using the Mgmt_NWK_Update_req command. This MAY be done if other failures have been reported or the network manager device itself has failures and a channel change MAY be desired. The network manager MAY request data from the list of devices that have reported interference plus other randomly selected routers in the network. The network manager SHOULD NOT request an update from the device that has just reported interference since this data is fresh already.
4. Upon receipt of the Mgmt_NWK_Update_notify, the network manager SHALL determine if a channel change is required using whatever implementation specific mechanisms are considered appropriate. The network manager device with just one channel allowed in the *apsChannelMask* parameter SHALL not issue the Mgmt_Nwk_Update_req command to request other devices to change the current channel. However, the network manager MAY report channel quality issues to the application.
5. If the above data indicate a channel change SHOULD be considered, the network manager completed the following:

6. Select a single channel based on the Mgmt_NWK_Update_notify based on the lowest energy. This is the proposed new channel. If this new channel does not have an energy level below an acceptable threshold, a channel change SHOULD NOT be done. Additionally, a new channel SHALL NOT belong to a PHY different from the one on which a network manager is operating now.
7. Prior to changing channels, the network manager SHOULD store the energy scan value as the last energy scan value and the failure rate from the existing channel as the last failure rate. These values are useful to allow comparison of the failure rate and energy level on the previous channel to evaluate if the network is causing its own interference.
8. The network manager SHOULD broadcast a Mgmt_NWK_Update_req notifying devices of the new channel. The broadcast shall be to all devices with RxOnWhenIdle equal to TRUE. The network manager is responsible for incrementing the *nwkUpdateId* parameter from the NIB and including it in the Mgmt_NWK_Update_req. The network manager SHALL set a timer based on the value of *apsChannelTimer* upon issue of a Mgmt_NWK_Update_req that changes channels and SHALL NOT issue another such command until this timer expires. However, during this period, the network manager can complete the above analysis. However, instead of changing channels, the network manager would report to the local application using Mgmt_NWK_Update_notify and the application can force a channel change using the Mgmt_NWK_Update_req.

Upon receipt of a Mgmt_NWK_Update_req with a change of channels, the local network manager SHALL set a timer equal to the *nwkNetworkBroadcastDeliveryTime* and SHALL switch channels upon expiration of this timer. Each node SHALL reset the total transmit count and the transmit failure counters, and copy the value from the NWK Update ID of the message into the *nwkUpdateId* of the NIB.

For devices with RxOnWhenIdle equals FALSE, any network channel change will not be received. On these devices or routers that have lost the network, an active scan SHALL be conducted on the *apsChannelMask* list in the APS IB using the extended PANID to find the network. If the extended PANID is found on different channels, the device SHOULD select the channel with the higher value in the *nwkUpdateId* parameter. If the extended PANID is not found using the *apsChannelMask* list, a scan SHOULD be completed using all channels within the current PHY.

This page intentionally left blank.

ANNEX F USAGE OF MULTIPLE FREQUENCY BANDS

F.1 Introduction

F.1.1 Scope

This annex clarifies uncertainties arising with Zigbee compliant devices that support several frequency bands.

F.1.2 Purpose

The Zigbee specification is based on the IEEE Std 802.15.4-2020 ([B1]) standard that defines multiple PHYs. A compliant device SHALL support at least one of the following options: O-QPSK PHY at 2.4 GHz frequency band or the BPSK PHY at both 868 MHz and 915 MHz bands or the FSK PHY located at 863-876MHz and 915-921MHz. Each of the frequency bands incorporates its own set of channels through a combination of channel numbers and channel pages. Additionally the following apply:

- A Zigbee compliant device declaring support of a frequency band SHALL support all the channels listed within the channel page for that frequency band.
- A Zigbee compliant device declaring support of the 868/915 MHz PHY SHALL support both 868 MHz and 915 MHz frequency bands within this PHY.

F.2 Channels and Channel Masks Management General Guideline

F.2.1 Channel Selection During Network Establishment

When there is a set of devices intended to be a part of the same Zigbee network, with devices of that set, potentially, supporting different frequency bands, the coordinator, during network establishment, MAY choose a channel from a frequency band that is not supported by some of the other devices.

Since, before a network is established, there is no mechanism for the coordinator to dynamically collect information about frequency bands supported on each and every device in the network, this issue MAY be categorized as a network commissioning issue and has to be resolved in the layers above the Zigbee stack's core.

In the case of multiband router interfaces that on network formation or joining do not join or form a network, these MAY be initialized using the NLME.NETWORK-ADD-INTERFACE command.

F.2.2 The Frequency Agility Feature Related Points

How a network manager or a device SHALL behave, considering the ability to support different frequency bands, is described in Annex E and in section 2.4.3.3.10 . Implementers of the frequency agility feature SHOULD take into account that it is prohibited for a network manager device to move a network from one PHY to another. This limitation is introduced in order to avoid the situations when a part of devices in the network cannot physically migrate to a channel from another PHY and therefore got lost. At the same time moving a network from one frequency band to another within 868/915 MHz PHY is allowed since support of both bands is mandatory in accordance with IEEE P802.15.4 (§C.7.2.3 [B1]). The application layer SHALL meet regional regulatory requirements by setting an appropriate value to the *apsChannelMaskList* parameter.

F.2.3 Network Management Services and Client Services Affected by Multiple Frequency Bands Support

The following Network Management Client Services and Network Management Services use the *ScanChannels* parameter and, therefore, have to be mentioned in regard of multiple frequency bands support: Mgmt_NWK_Disc_req, Mgmt_NWK_Update_req and NLME-JOIN.request. In case the *ScanChannels* bitmask includes a channel(s) from unsupported frequency band the INVALID_PARAMETER (see [B1]) error status is supposed to be raised from the MAC layer to the NWK layer. If the destination addressing mode in the Mgmt_NWK_Disc_req and

Mgmt_NWK_Update_req commands was unicast then the Remote Device SHALL incorporate the error status into the status field of the correspondent Mgmt_NWK_Disc_rsp and Mgmt_NWK_Update_notify commands. The same error status shall be reported in NLME-JOIN.confirm primitive sent in response to an NLME-JOIN.request primitive if the latter contains unsupported channels.

In case the NLME-JOIN.request primitive is used by the application layer to request a device to switch to a new channel (the *RejoinNetwork* parameter is equal to 0x03) then the application layer, by implementation-specific means, has to ensure that the chosen channel is supported by all other devices in the network, to avoid the situation when some of the devices might be lost from the network due to inability to switch to an unsupported channel.

F.3 Timing Issues

Different frequency bands declared in the IEEE Std 802.15.4 2015 standard provide different bit rates. Therefore the Zigbee stack's time-related parameters have to be adjusted accordingly to achieve the stable operation on each of the supported frequency bands. The Zigbee stack's time-related parameters can be divided in two groups in regard of multiple frequency bands support: the first group includes time-related parameters that have a direct impact on the Zigbee stack's core's functioning and that ensure that the core's functioning is correct; the second group consists of the time-related parameters that have to be configured by an application. The Zigbee specification controls the first group of parameters and declares them in a way that makes them dependent on the currently used frequency band. These parameters are presented in Table F-1 and their values SHALL be updated automatically each time a device migrates from one frequency band to another.

Table F-1. Internal Time-related Parameters

Parameter	**Reference**
:Config_NWK_Time_btwn_Scans	Section 2.5.5.1, Table 2-135
nwkcRouteDiscoveryTime	Section 3.5.1, Table 3-61
nwkcMaxBroadcastJitter	Section 3.5.1, Table 3-61
nwkcRREQRetryInterval	Section 3.5.1, Table 3-61
nwkcMinRREQJitter	Section 3.5.1, Table 3-61
nwkcMaxRREQJitter	Section 3.5.1, Table 3-61
nwkPassiveAckTimeout	Section 3.5.2, Table 3-62
nwkNetworkBroadcastDeliveryTime	Section 3.5.2, Table 3-62
apsSecurityTimeOutPeriod	Section 4.4.12, Table 4-35

This page intentionally left blank.

ANNEX G INTER-PAN COMMUNICATIONS

G.1 Scope and Purpose

This annex defines a mechanism whereby Zigbee devices can perform exchanges of information with devices in their local area without having to form or join the same Zigbee network. This capability is used in a number of Zigbee functions from extending Smart Energy networks to simple low cost devices, for Green Power end devices, or for Touchlink commissioning.

G.2 General Description

G.2.1 What Inter-PAN APS Does

A schematic view of the Zigbee stack enabling Inter-PAN data and Green Power Device Frame exchange is shown in Figure G-1.

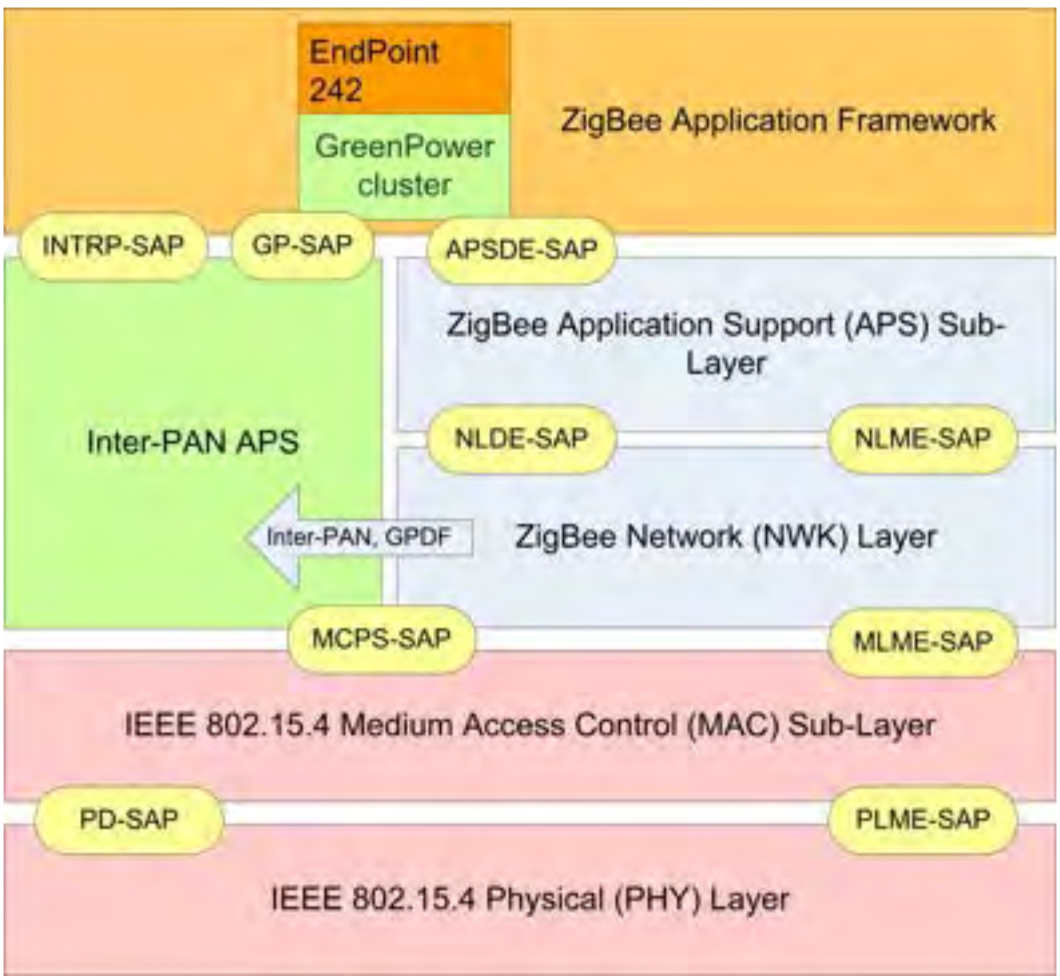


Figure G-1. Zigbee Stack with Inter-PAN APS

All features relating to Green Power have been removed from this section since that is now covered elsewhere. Refer to [B5] for the relevant Green Power specification details.

Inter-PAN data exchanges and Green Power Device Frame (GPDF) exchanges are handled by a special "stub" of the Application Support Sub-Layer, which is accessible through a special Service Access Point (SAP), the INTRP-SAP, parallel to the APSDE-SAP. The Inter-PAN data exchange architecture is used by several different mechanisms within Zigbee devices.

The same Inter-PAN APS is intended to be used for these different services even if how they use it varies slightly. In case of Inter-PAN data exchanges, the Inter-PAN APS performs just enough processing to pass application data frames to the MAC for transmission and to pass Inter-PAN application frames from the MAC to the application on receipt. In case of Green Power Device Frame exchanges, the Inter-PAN APS also performs security processing of incoming and outgoing GPDF, as well as buffering of outgoing GPDF (see [B5]). The incoming GPDF are delivered to the application on endpoint 242 and handled by that; see the specification of the Green Power cluster residing on endpoint 242 [B4].

G.2.2 Service Specification

The INTRP-SAP is a data service comprising eight primitives.

- INTRP-DATA.request - Provides a mechanism for a sending device to request transmission of an Inter-PAN message.
- INTRP-DATA.confirm - Provides a mechanism for a sending device to understand the status of a previous request to send an Inter-PAN message.
- INTRP-DATA.indication - Provides a mechanism for identifying and conveying an Inter-PAN message received from a sending device.

G.2.3 The INTRP-DATA.request Primitive

The INTRP-DATA.request primitive allows an application entity to request data transmission via the Inter-PAN APS.

G.2.3.1 Semantics of the Service Primitive

```
INTRP-DATA.request          {
                            SrcAddrMode
                            DstAddrMode
                            DstPANId
                            DstAddress
                            ProfileId
                            ClusterId
                            ASDULength
                            ASDU
                            ASDUHandle
                        }
```

Table G-1 specifies the parameters of the INTRP-DATA.request primitive.

Table G-1. Semantics of the INTRP-DATA.request Primitive

Name	Type	Valid Range	Description
SrcAddrMode	Integer	0x00 – 0x03	The source addressing mode for the MPDU to be sent. This value can take one of the following values: 0 x 00 = no address (SrcPANId and SrcAddress omitted). 0 x 01 = reserved. 0 x 02 = 16 bit short address. 0 x 03 = 64 bit extended address.
DstAddrMode	Integer	0x01 – 0x03	The addressing mode for the destination address used in this primitive. This parameter can take one of the values from the following list: 0x01 = 16-bit group address 0x02 = 16-bit NWK address, usually the broadcast address 0xffff 0x03 = 64-bit extended address
DstPANID	16-bit PAN ID	0x0000 – 0xFFFF	The 16-bit PAN identifier of the entity or entities to which the ASDU is being transferred or the broadcast PANId 0xffff.
DstAddress	16-bit or 64-bit address	As specified by the AddrMode parameter	The address of the entity or entities to which the ASDU is being transferred.
ProfileId	Integer	0x0000 – 0xffff	The identifier of the application profile for which this frame is intended.
ClusterId	Integer	0x0000 – 0xffff	The identifier of the cluster, within the profile specified by the ProfileId parameter, which defines the application semantics of the ASDU.
ASDULength	Integer	0x00 – (*aMax-MACFrameSiz e* - 9)	The number of octets in the ASDU to be transmitted.
ASDU	Set of octets	-	The set of octets forming the ASDU to be transmitted.
ASDUHandle	Integer	0x00 – 0xff	An integer handle associated with the ASDU to be transmitted.

G.2.3.2 When Generated

This primitive is generated by the local application entity when it wishes to address a frame to one or more peer application entities residing on neighboring devices using Inter-PAN data.

G.2.3.3 Effect on Receipt

On receipt of the INTRP-DATA.request primitive by the Inter-PAN APS, the Inter-PAN APS will construct and transmit an Inter-PAN frame. This frame SHALL have a Protocol Version sub-field and the Frame Type sub-field of the NWK Frame Control field set to the values as specified in section G.3.2.1. The frame SHALL contain the given ASDU and set the parameters using the MCPS-DATA.request primitive of the MAC sub-layer, as described in section G.3.1.1. Once the corresponding MCPS-DATA.confirm primitive is received, the stack SHALL generate the INTRP-DATA.confirm primitive with a status value reflecting the status value returned by the MAC.

G.2.4 The INTRP-DATA.indication Primitive

The INTRP-DATA.indication primitive allows the Inter-PAN APS to inform the next higher layer that it has received a frame that was transmitted via the Inter-PAN APS on another device.

G.2.4.1 Semantics of the Service Primitive

The primitive interface is as follows:

```
INTRP-DATA.indication         {
                              SrcAddrMode
                              SrcPANId
                              SrcAddress
                              DstAddrMode
                              DstPANId
                              DstAddress
                              ProfileId
                              ClusterId
                              ASDULength
                              ASDU
                              LinkQuality
                          }
```

Table G-2 defines the parameters of the INTRP-DATA.indication primitive.

Table G-2. Parameters of the INTRP-DATA.indication Primitive

Name	Type	Valid Range	Description
SrcAddrMode	Integer	0x00- 0x03	The source addressing mode for the MPDU to be sent. The following values are allowed: 0x00 – no address (SrcPANId and SrcAddress omitted) 0x01 = reserved 0x02 = 16 bit short address 0x03 = 64 bit extended address
SrcPANId	16-bit PAN Id	0x0000 – 0xffff	The 16-bit PAN identifier of the entity from which the ASDU is being transferred.
SrcAddress	64-bit address	As specified by the	The device address of the entity from which the ASDU is being transferred.

Name	Type	Valid Range	Description
		SrcAddrMode parameter	
DstAddrMode	Integer	0x00 – 0x03	The addressing mode for the destination address used in this primitive. This parameter can take one of the values from the following list: 0x00 = no address (DstPANId and DstAddr omitted) 0x01 = 16-bit group address 0x02 = 16-bit NWK address, usually the broadcast address 0xffff 0x03 = 64-bit extended address
DstPANID	16-bit PAN Id	0x0000 – 0xffff	The 16-bit PAN identifier of the entity or entities to which the ASDU is being transferred or the broadcast PAN ID 0xffff.
DstAddress	16-bit or 64-bit address	As specified by the DstAddrMode parameter	The address of the entity or entities to which the ASDU is being transferred.
ProfileId	Integer	0x0000 – 0xffff	The identifier of the application profile for which this frame is intended.
ClusterId	Integer	0x0000 – 0xffff	The identifier of the cluster, within the profile specified by the ProfileId parameter, which defines the application semantics of the ASDU.
ASDULength	Integer	0x00 – (*aMaxMACFrameSize* - 9)	The number of octets in the ASDU to be transmitted.
ASDU	Set of octets	-	The set of octets forming the ASDU to be transmitted.
LinkQuality	Integer	0x00 – 0xff	The link quality observed during the reception of the ASDU.

G.2.4.2 When Generated

This primitive is generated and passed to the application in the event of the receipt, by the Inter-PAN APS, of a MCPS-DATA.indication primitive from the MAC sub-layer, containing a frame that was generated by the Inter-PAN APS of a peer Zigbee device, and that was intended for the receiving device.

G.2.4.3 Effect on Receipt

Upon receipt of this primitive the application is informed of the receipt of an application frame transmitted, via the Inter-PAN APS, by a peer device and intended for the receiving device. The values of the INTRP-DATA.indication SHALL be copied into the matching field names of the APSDE-DATA.indication. Additionally these fields SHALL be set as follows:

1. The DstAddrMode SHALL be set to 0x04.
2. The DstAddress SHALL be set to the DstAddress of the INTRP-DATA.indication primitive.
3. The SrcAddrMode SHALL be set to 0x04.
4. The SrcAddress SHALL be set to the SrcAddress of the INTRP-DATA.indication primitive.
5. The SecurityStatus field enumeration SHALL be set to UNSECURED.
6. The Inter-PAN field SHALL be set to TRUE.

G.2.5 Qualifying and Testing of Inter-PAN Messages

Support for Inter-PAN messages and Green Power is optional. If a device claims Inter-PAN communication support then certification and application level testing SHALL ensure both the sending and receiving devices correctly react and understand the INTRP-DATA.request and INTRP-DATA.indication primitives. Green Power certification and application level testing SHALL also ensure the GP-DATA.request, GP-DATA.indication, GP-SEC.request, and GP-SEC.response primitives are supported as mandated by the Green Power Specification [B4].

G.3 Frame Formats

The overall view of a Zigbee frame is as shown in Figure G-2.

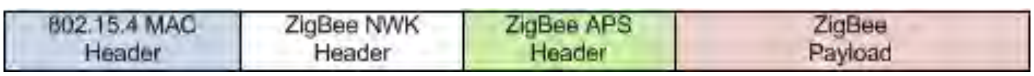


Figure G-2. Zigbee Frame Format Overview

Briefly, the frame contains the familiar headers controlling the operation of the MAC sub-layer, the NWK layer and the APS. Following these, there is a payload, formatted as specified in [B1].

Since most of the information contained in the NWK header is not relevant for Inter-PAN transmission, the Inter-PAN frame, shown in Figure G-3, contains only a stub of the NWK header. A Inter-PAN APS header is also used and is described in section G.2.3.3.

Octets: 2	**1**	**variable**	**2**
Frame Control	Sequence Number	Addressing Fields	NWK Frame Control
802.15.4 MAC Header			NWK Header

Figure G-3. Inter-PAN Frame Format

For Green Power Device Frames there is a different set of MAC and NWK headers as shown in Figure G-4.

Octets: 2	1	4/10/12/ Variable	1	0/1	0/4	0/4	Variable	0/2/4
Frame Control	Sequence Number	Addressing Fields	NWK Frame Control	Extended NWK Frame Control	GPD SrcID	Security Frame Counter	GPDF Application Payload	MIC
802.15.4 MAC Header			GPDF NWK Header				GPDF Application Payload	GPDF NWK Trailer

Figure G-4. Green Power Device Frame Format

G.3.1 MAC Header

The 802.15.4 MAC header has several options depending on how the frame is being used. The MAC header fields are shown in 3 with notes on their use.

Table G-3. MAC Header Fields for Inter-PAN APS Frames

Field Name	Octets	Usage
Frame Control	2	Varies by Inter-PAN APS frame
Sequence Number	1	Normally used as MAC sequence number, increasing for each frame sent. Green Power usage discussed in GreenPower Specification reference [B4].
Destination PAN ID	0/2	May be set as the PANID of the destination or 0xffff.
Destination Address	2/8	Normally either broadcast short address or a 64 bit long address of the destination. Green Power usage discussed in GreenPower Specification reference [B4].
Source PAN ID	0/2	Used in Inter-PAN messaging but not in Green Power Device Frames.
Source Address	2/8	Normally set to the 64 bit address of the source device. Green Power usage discussed in GreenPower Specification reference [B4].

The MAC header usage varies by application using the Inter-PAN messaging.

G.3.1.1 MAC Header Usage for Inter-PAN Messaging

Because Inter-PAN messaging is used for devices not on the Zigbee network, short addressing is not normally used unless it is the broadcast short address such that any device within range can respond. Otherwise the 64 bit long addresses are used for source and destination addressing. Source and Destination PANID's MAY be used or MAY be omitted.

G.3.2 Network Header

G.3.2.1 Stub NWK Header for Inter-PAN Messages

The stub NWK Header for Inter-PAN messages is shown below in Figure G-5.

Octets:2
NWK Frame Control

Figure G-5. Stub NWK Header for Inter-PAN messages

The NWK header Frame control field for the Inter-PAN messages is formatted exactly as the NWK header used by other Zigbee frames, see section 3.3.1.1 of the current specification.

For Inter-PAN messages, the frame type 0b11 is used with the protocol version of the Zigbee stack. All other sub-fields SHALL have a value of 0.

G.3.2.1.1Remaining Fields of the Stub NWK Header for GPDF

The GPDSrcID field is present if the FrameType sub-field is set to 0b00 and the ApplicationID sub-field of the Extended NWK Frame Control field is set to 0b000 (or not present). It is also present if the FrameType sub-field is set to 0b01, the NWK Frame control Extension sub-field is set to 0b1, and the ApplicationID sub-field of the Extended NWK Frame Control field is set to 0b000. The GPDSrcID field carries the unique identifier of the GPD, to/by which this GPDF is sent. The value of 0x00000000 indicates unspecified. The value of 0xffffffff indicates all. The values 0xfffffff9 – 0xfffffffe are reserved. The GPDSrcID field is not present if the FrameType sub-field is set to 0b01 and the Extended NWK Frame control sub-field is set to 0b0. Unique identification of the GPD by an address is not required then. The GPDSrcID field is not present if the ApplicationID sub-field of the Extended NWK Frame Control field is set to 0b010. The GPD is then identified by its IEEE address, which is then carried in the corresponding MAC address field, source or destination for the GPDF sent by or to the GPD, respectively.

The presence and length of the Security frame counter field is dependent on the value of ApplicationID and SecurityLevels sub-field, as described above.

The MIC field carries the Message Integrity Code for this message, calculated as specified in section A.1.4 of the current GreenPower Specification [B4]. Its presence and length is dependent on the value of ApplicationID and SecurityLevel sub-fields, as described above.

The application payload of the GPDF is defined in [B4], section A.1.4.1.6.

G.3.3 Inter-PAN APS Header

The format of the Inter-PAN APS header is shown in Figure G-6. This is used in normal Inter-PAN messages and Touchlink messages but not in Green Power Device Frames.

Octets: 1	0/2	2	2
APS frame control	Group address	Cluster identifier	Profile identifier
	Addressing fields		

Figure G-6. Inter-PAN APS Header Format

The Inter-PAN APS header contains only 4 fields totaling a maximum of 7 octets in length.

The APS frame control field SHALL be 1 octet in length and is identical in format to the frame control field of the general APDU frame in [B3] (see Figure G-7).

Bits: 0-1	2-3	4	5	6	7
Frame type	Delivery Mode	Reserved	Security	ACK request	Extended Header Present

Figure G-7. Format of the APS Frame Control Field for Inter-PAN Messages

The fields of the frame control field have the following values:

- The frame type sub-field SHALL have a value of 0b11, which is the Inter-PAN APS frame type.
- The delivery mode sub-field MAY have a value of 0b00, indicating unicast, 0b10, indicating broadcast or 0b11 indicating group addressing.
- Security is never enabled for Inter-PAN transmissions. This sub-field SHALL be a value of 0.
- The ACK request sub-field SHALL have a value of 0, indicating no ACK request. No APS ACKs are to be used with Inter-PAN transmissions.
- The extended header present sub-field SHALL always have a value of 0, indicating no extended header.

The optional group address shall be present if and only if the delivery mode field has a value of 0x0b11. If present it SHALL contain the 16-bit identifier of the group to which the frame is addressed.

The cluster identifier field is 2 octets in length and specifies the identifier of the cluster to which the frame relates and which SHALL be made available for filtering and interpretation of messages at each device that takes delivery of the frame. For touchlink this has a value of 0x1000.

The profile identifier is two octets in length and specifies the Zigbee profile identifier for which the frame is intended and SHALL be used during the filtering of messages at each device that takes delivery of the frame. For touchlink this has the value of 0xc05e.

G.4 Frame Processing

Assuming the INTRP-SAP described above, frames transmitted using the Inter-PAN APS are processed as described here.

G.4.1 Inter-PAN Transmission (non Green Power Device Frames)

On receipt of the INTRP-DATA.request primitive, the Inter-PAN APS SHALL construct a Inter-PAN APS frame. The header of the Inter-PAN APS frame SHALL contain a NWK and an APS frame control field as described in section G.3, a cluster identifier field equal to the value of the ClusterId parameter of the INTRP-DATA.request and a profile identifier field equal to the value of the ProfileId parameter. If the DstAddrMode parameter of the INTRP-DATA.request has a value of 0x01, indicating group addressing, then the APS header SHALL also contain a group address field with a value corresponding to the value of the DstAddress parameter. The payload of the Inter-PAN APS frame SHALL contain the data payload to be transmitted.

The Inter-PAN APS frame will then be transmitted using the MCPS-DATA.request primitive of the MAC sub-layer with key primitive parameters set as follows:

- The value of the SrcAddrMode parameter of the MCPS-DATA.request SHALL always be set to a value of three, indicating the use of the 64-bit extended address.
- The SrcPANId parameter SHALL be equal to the value of the macPANID attribute of the MAC PIB.
- The SrcAddr parameter SHALL always be equal to the value of the MAC sub- layer constant aExtendedAddress.
- If the DstAddrMode parameter of the INTRP-DATA.request primitive has a value of 0x01, then the DstADdrMode parameter of the MCPS-DATA.request SHALL have a value of 0x02. Otherwise, the DstAddrMode parameter of the MCPS-DATA.request SHALL reflect the value of the DstAddrMode parameter of the INTRP-DATA.request.
- The DstPANId parameter SHALL have the value given by the DstPANID parameter of the INTRP-DATA.request primitive.
- If the DstAddrMode parameter of the INTRP-DATA.request has a value of 0x01, indicating group addressing, then the value of the DstAddr parameter of the MCPS-DATA.request shall be the broadcast address 0xffff.

Otherwise, value of the DstAddr parameter SHALL reflect the value of the DstAddress parameter of the INTRP-DATA.request primitive.

- The MsduLength parameter SHALL be the length, in octets, of the Inter-PAN APS frame.
- The Msdu parameter SHALL be the Inter-PAN APS frame itself.
- If the transmission is a unicast, then the value of the TxOptions parameter shall be 0x01, indicating a request for acknowledgement. Otherwise, the TxOptions parameter SHALL have a value of 0x00, indicating no options.

On receipt of the MCPS-DATA.confirm primitive from the MAC sub-layer, the Inter-PAN APS will invoke the INTRP-DATA.confirm primitive with a status reflecting the status returned by the MAC.

G.4.2 Inter-PAN Reception (non Green Power Device Frames)

On receipt of the MCPS-DATA.indication primitive from the MAC sub-layer, the receiving entity - in case of a Zigbee device this is normally the NWK layer - SHALL determine whether the frame SHOULD be passed to the Inter-PAN APS or processed as specified in [B5]. For a frame that is to be processed by the Inter-PAN APS, the non- varying sub-fields of the NWK frame control field SHALL be set exactly as described in section G.3.2.1 and the APS frame control field SHALL be set exactly as described in section G.3.3. Any variation from this format SHALL trigger the message to be dropped and no further processing SHALL be done.

If the delivery mode sub-field of the APS frame control field of the Inter-PAN APS header has a value of 0b11, indicating group addressing, then, if the device implements group addressing, the value of the group address field SHALL be checked against the NWK layer group table, and, if the received value is not present in the table, the frame SHALL be discarded with no further processing or action.

On receipt of a frame for processing, the Inter-PAN APS SHALL generate an INTRP- DATA.indication with parameter values as follows:

- The value of the SrcAddrMode parameter of the INTRP-DATA.indication SHALL always be set to a value of three, indicating the use of the 64-bit extended address.
- The value of the SrcPANId parameter SHALL reflect that of the SrcPANId parameter of the MCPS-DATA.indication.
- The SrcAddress parameter of the INTRP-DATA.indication SHALL always reflect the value of a 64-bit extended address.
- Values for the DstAddrMode parameter SHALL be one of:
  - 0x03, if the DstAddrMode parameter of the INTRP-DATA.indication has a value of 0x03.
  - 0x02, if the DstAddrMode parameter of the INTRP-DATA.indication has a value of 0x02
- The value of the DstPANId parameter of the INTRP-DATA.indication SHALL reflect the value of the DstPANId parameter of the MCPS-DATA.indication.
- If the DstAddrMode parameter of the INTRP-DATA.indication has a value of 0x01, indicating group addressing then the DstAddress parameter of the INTRP-DATA.indication SHALL reflect the value of the group address field of the Inter-PAN APS header. Otherwise, the value of the DstAddress parameter of the INTRP-DATA.indication SHALL reflect the value of the DstAddr parameter of the MCPS-DATA.indication.
- The value of the ProfileId parameter SHALL be the same as the value of the profile identifier field of the Inter-PAN APS header.
- The value of the ClusterId parameter SHALL be the same as the value of the cluster identifier field of the Inter-PAN APS header.
- The ASDULength field SHALL contain the number of octets in the Inter-PAN APS frame payload.
- The ASDU SHALL be the Inter-PAN APS payload itself.

- The value of the LinkQuality parameter SHALL reflect the value of the mpduLinkQuality parameter of the MCPS-DATA.indication.

G.5 Inter-PAN Best Practices

Network Channel Manager Inter-PAN support is not specified in Annex E of the core stack specification ([B3]). New channel notifications will not be broadcast Inter-PAN. Inter-PAN devices which do not receive the network channel change will need to perform the network discovery procedure described in section 3.6.1.5.1.

It is recommended that devices that use Inter-PAN SHOULD implement an allow list of known accepted commands and constrain the list to only the necessary commands. Inter-PAN commands SHOULD carefully screened by the receiving device since they can be sent by devices that do not have network security credentials and are performing an active attack.

ANNEX H SECURITY TEST VECTORS FOR GREEN POWER DEVICE FRAMES

This section has been entirely superseded by the Green Power Specification 14-0563-16 [B4].

ANNEX I ZIGBEE TLV DEFINITIONS AND FORMAT

This annex describes the definition and format of Zigbee Type-Length-Value data elements (TLVs). These definitions are specific to the Zigbee Core specification. The advantage of TLV formatted data is that new data fields can be packed into messages in a future proof way that allows a device to parse as much data as it understands regardless of where the unknown data appears. TLVs can be declared as mandatory or optional independent of the order they are in.

I.1 General Format

The general format of TLVs is defined in Table I-1.

Table I-1. General TLV Format

Tag Field	Length Field	Value Field
1-byte	1-byte	Variable Length (Length Field Value + 1)

The actual size of the Value field is always interpreted as the value in the received Length field + 1.

I.1.1 Reserved Fields

All reserved TLV fields SHALL be set to 0 on transmission and ignored on reception.

I.1.2 Tag Id Ranges

Zigbee TLV Tags are divided into locally and globally scoped IDs. Locally scoped tag IDs have a format specific to a particular message and MAY overlap with Local tag IDs of different messages. Request and response messages SHALL have separate local tag IDs. Local tag ID formats are defined in the section of the Zigbee spec where they are sent. They share the same general message overhead as described in this Annex.

Global tag IDs have a single format across all messages and layers and are used in 2 or more different messages. In general, Global Tag IDs represent a global state and is not completely ephemeral in nature. However, not all global IDs have meaning for all messages and a known global tag ID MAY still trigger the receiving device to reject the message. This will be described in the handling of a particular message.

The tag ID Ranges are defined in Table I-2.

Table I-2. Tag ID Ranges

Tag Range	Scope
0 – 63	*Local*
64-255	Global

I.2 Rules for TLVs in Message

I.2.1 Order

When a message contains TLVs, order SHALL NOT matter. TLVs MAY be concatenated in any order by the sender. No functional behavior SHALL be required of the receiver based on the order of multiple TLVs in a single message.Duplication of Tag Ids.

Duplicate TLV Tag IDs in messages are NOT allowed with one exception, described below. When multiple TLVs with the same ID are found in a single message the message SHALL be rejected and dropped.

The Manufacturer Specific Global TLV is the one exception and that TLV MAY occur multiple times in a message. Interpretation and behavior on receipt of any Manufacturer Specific Global TLV is outside the scope of this specification. Devices that receive a Manufacturer Specific Global TLV with content they do not understand SHALL treat this as an Unknown Tag as described in the rules for Unknown Tags.

I.2.2 Global Tags

In general, all Global Tags SHALL be accepted for all messages. If the message processing description in this section does not have specific rules for a Global TLV then the received Global TLVs are silently ignored. The processing rules of a message MAY override this behavior and describe what to do when one or more known Global TLVs are unexpectedly received.

Global TLVs MAY be appended anywhere but are only actionable when the message processing description dictates how they are handled. NIB, AIB, or other state SHALL not be updated unless the local message processing explicitly mentions that behavior.

I.2.3 Unknown Tags

An Unknown Tag is defined as one where the Tag ID is neither a defined global ID nor is the ID a Local ID defined for the specific command or datagram containing the TLV.

Receiving devices SHALL ignore all unknown Tag IDs and no processing SHALL be done for them.

I.2.4 Extending Existing TLVs

Existing TLVs MAY be extended in a future version of this specification. By default, known TLVs that are longer than their defined minimum length SHALL NOT be dropped. Known elements in the TLV shall be interpreted and the unknown elements ignored.

When a receiver SHALL store TLVs, the receiver SHALL also store the full extended TLV.

I.2.5 Malformed TLVs

When TLVs are smaller than the minimum size defined in the specification they SHALL be considered malformed. Malformed TLVs are also defined as the case where the size is within the minimum and maximum length, but the data contained has been truncated. For example, if the TLV defines a list of 2-byte node IDs and the length value causes the TLV to truncate in the middle of a 2-byte node ID, this is considered malformed.

Malformed TLVs SHALL generate an error. Processing of the message SHALL terminate and all TLVs contained in the message SHALL NOT be processed, including all TLVs preceding the malformed TLV.

I.2.6 Encapsulation TLVs

A select few TLVs are allowed to contain other TLVs. They are called Encapsulation TLVs. Encapsulation TLVs SHALL NOT contain Encapsulation TLVs.

To validate an Encapsulation TLV the General TLV Processing in Section I.4.8 SHALL be executed on the TLVs inside the Encapsulation TLV with one additional requirement. If an Encapsulation TLV contains another Encapsulation TLV the outer Encapsulation TLV SHALL be considered malformed and no further process SHALL be done.

I.2.7 General TLV Processing

1. Examine all TLVs in the message.
2. Check for any duplicate TLV tag IDs.
    a. If any TLV ID is duplicated, except for the Manufacturing Specific TLV, do the following:
        i. The message shall be rejected. No further processing SHALL be done.
    b. Otherwise, continue processing.
3. Determine if any TLVs are malformed.

   a. Examine all known TLVs.
   b. If any TLV is less than the minimum required length, then the message SHALL be rejected. A status of INVALID_TLV SHALL be returned to the calling routine and no further processing by this routine SHALL be done.
   c. If the TLV length is greater than the minimum but causes the known data to be truncated, then the message SHALL be rejected. A status of INVALID_TLV SHALL be returned to the calling routine and no further processing by this routine SHALL be done.
   d. If the TLV is an encapsulation TLV and is contained inside another Encapsulation TLV, the outer Encapsulation TLV is considered invalid. A status of INVALID_TLV SHALL be returned to the calling routine and no further processing by this routine SHALL be done.
   e. If the TLV is an Encapsulation TLV and is not inside another Encapsulation TLV, execute the rules in this section (I.4.8) to validate the TLVs contained inside.
      i. If the result indicates that the Encapsulation TLV is malformed then that TLV SHALL be discarded.
   f. Otherwise, continue processing.
4. General TLV Processing has succeeded, return SUCCESS to the calling routine.

Refer to the message specific processing rules to determine whether the message will be accepted.Known Data being truncated

All TLVs MAY be extended in future versions of the spec. The minimum length of a TLV is the length for the TLV when it was first introduced in the specification. Extensions to the TLV in later versions of the specification SHALL NOT change the minimum length. However, a received TLV can be longer than the minimum length and still be considered invalid if the TLV has been extended with a new field of a defined length. For example, a TLV that has an EUI64 in a prior version of the spec and was extended to include a Node ID field. The minimum length of that TLV will always be 8 bytes. If the node knows about the newer version of the TLV that includes a Node ID but receives only 9 bytes, it rejects the entire TLV. A node that does not know about the node ID field would end up accepting the 9 bytes but only processing the EUI64 field.

For TLVs with sets or lists of parameters the receiver is required to validate that the list is the appropriate length. For example if a TLV has a 1-byte count field indicating a number of Node IDs (2-bytes) that follow, if the overall length would cause it to truncate the last node ID the whole TLV is rejected. Additionally, if there is a count field within the TLV and the value of count is more than the number of items in the TLV the TLV SHALL be considered invalid. For example, a TLV with a value length of 7 containing a 1-byte count field that indicates 8 items of 1-byte each follow, would be considered invalid. The TLV value length SHOULD have been 9 (1-byte count field followed by 8-bytes of items).

I.3 Security

TLVs are secured by the mechanism that transports them. It is up to the commands and messages that transport them to dictate any special security processing required for transmission and reception.

I.4 Global TLV IDs

Table I-3 defines the global TLV IDs defined by the Zigbee specification.

Table I-3. Global TLV Definitions

TLV Tag ID	Minimum Length (Bytes)	Name
64	2	Manufacturer Specific Global TLV
65	2	Supported Key Negotiation Methods Global TLV
66	4	PAN ID Conflict Report Global TLV
67	2	Next PAN ID Global TLV
68	4	Next Channel Change Global TLV
69	16	Symmetric Passphrase Global TLV
70	2	Router Information Global TLV
71	2	Fragmentation Parameters Global TLV
72		Joiner Encapsulation Global TLV
73		Beacon Appendix Encapsulation Global TLV
74		Reserved
75		Configuration Parameters Global TLV
76		Device Capability Extension Global TLV (Refer to the Zigbee Direct specification for more details.)
77-255		Reserved

I.4.1 Manufacturer Specific Global TLV (ID 64)

The Manufacturing Specific Global TLV defines a TLV that is defined outside the Zigbee specification. Its interpretation and handling are outside the scope of this specification. The Manufacture Specific Global TLV SHALL be at least 2 bytes in length indicating the Zigbee Manufacturer ID. The format is specified in Figure I-1.

The Manufacturer associated with this Manufacture ID will define the additional data in the Manufacturing Specific Global TLV. Other manufacturers MAY make use of those TLVs with a different Manufacturer's ID.

The Manufacturer Specific Global TLV MAY be appended on the end of any command frame.

Octets: 2	Varies
Zigbee Manufacturer ID	Additional Data

Figure I-1. Manufacturer Specific Global TLV Data

I.4.2 Supported Key Negotiation Methods Global TLV (ID 65)

This TLV defines the set of Key Negotiation protocols that the sending device supports. When sent as part of an IEEE Std 802.15.4 Beacon it indicates the set of Key Negotiation Protocols that the Trust Center Supports.

The format of this TLV is defined in Figure I-2.

Octets: 1	Octets: 1	Octets: 8
Key Negotiation Protocols Bitmask	Pre-shared Secrets Bitmask	Source Device EUI64

Figure I-2. Supported Key Negotiation Methods Global TLV Data

The Key Negotiation Protocols Bitmask is defined in Table I-4.

Table I-4. Key Negotiation Protocols Bitmask

Bit	Description
0	Static Key Request (Zigbee 3.0 Mechanism)
1	SPEKE using Curve25519 with Hash AES-MMO-128
2	SPEKE using Curve25519 with Hash SHA-256
3 – 7	Reserved

The Pre-shared Secrets Bitmask is defined in Table I-5.

Table I-5. Supported Pre-shared Secrets Bitmask

Bit	Name	Description
0	Symmetric Authentication Token	This is a token unique to the Trust Center and network that the device is running on, and is assigned by the Trust center after joining. The token is used to renegotiate a link key using the Key Negotiation protocol and is good for the life of the device on the network.
1	Install Code Key	128-bit pre-configured link-key derived from install code
2	Passcode Key	A variable length passcode for PAKE protocols. This passcode can be shorter for easy entry by a user.
3	Basic Access Key	This key is used by other Zigbee specifications for joining with an alternate pre-shared secret. The definition and usage is defined by those specifications. The usage is optional by the core Zigbee specification.
4	Administrative Access Key	This key is used by other Zigbee specifications for joining with an alternate pre-shared secret. The definition and usage is defined by those specifications. The usage is optional by the core Zigbee specification.
5-7	Reserved	-

I.4.3 PAN ID Conflict Report Global TLV (ID 66)

This TLV is 2-bytes in length and indicates the next channel that will be used once a Network Update command is received to change PAN IDs.

This TLV defines information about PAN ID Conflicts detected by the coordinator and routers on the network. This TLV contains the NIB value of *nwkPanIdConflictsCount* as formatted in Figure I-3.

Octets:2
nwkPanIdConflictCount

Figure I-3. PAN ID Conflict Global TLV

I.4.4 Next PAN ID Change Global TLV (ID 67)

This TLV is 2-bytes in length and indicates the next channel that will be used once a Network Update command is received to change PAN IDs.

I.4.5 Next Channel Change Global TLV (ID 68)

This TLV is 4-bytes in length and indicates the next channel that will be used once a start channel change command is received. The format is defined according to the Channels Field bitmap defined in Table 3-7. Only 1 channel page and channel bit SHALL be set.

I.4.6 Symmetric Passphrase Global TLV (ID 69)

This TLV is 16-bytes in length and indicates a 128-bit Symmetric Passphrase Global TLV.

I.4.7 Router Information Global TLV (ID 70)

This TLV is 2-bytes in length and is a bitmask indicating data about the local router. Table I-6 defines the bits.

Table I-6. Router Information Global TLV Bitmap Definitions

Bit	Name	Description
0	Hub Connectivity	This bit indicates the state of *nwkHubConnectivity* from the NIB of the local device. It advertises whether the router has connectivity to a Hub device as defined by the higher-level application layer. A value of 1 means there is connectivity, and a value of 0 means there is no current Hub connectivity.
1	Uptime	This 1-bit value indicates the uptime of the router. A value of 1 indicates the router has been up for more than 24 hours. A value of 0 indicates the router has been up for less than 24 hours.
2	Preferred Parent	This bit indicates the state of *nwkPreferredParent* from the NIB of the local device. When supported, it extends Hub Connecivity, advertising the devices capacity to be the parent for an additional device. A value of 1 means that this device should be preferred. A value of 0 indicates that it should not be preferred. Devices that do not make this determination SHALL always report a value of 0.
3	Battery Backup	This bit indicates that the router has battery backup and thus will not be affected by temporary losses in power.
4	Enhanced Beacon Request Support	When this bit is set to 1, it indicates that the router supports responding to Enhanced beacon requests as defined by IEEE Std 802.15.4. A zero for this bit indicates the device has no support for responding to enhanced beacon requests.
5	MAC Data Poll Keepalive Support	This indicates that the device has support for the MAC Data Poll Keepalive method for End Device timeouts.
6	End Device Keepalive Support	This indicates that the device has support for the End Device Keepalive method for End Device timeouts.
7	Power Negotiation Support	This indicates the device has support for Power Negotiation with end devices.
8-15	Reserved	These bits SHALL be set to 0.

I.4.8 Fragmentation Parameters Global TLV (ID 71)

This TLV specifies the maximum reassembled input buffer size of the associated node. The Reassembled Buffer Size includes the fragmentation capabilities of the device and thus would be larger than the normal MPDU of the underlying NWK and MAC layers.

Octets: 2	1	2
Node ID	Fragmentation Options	Maximum Incoming Transfer Unit

Figure I-4. Fragmentation Parameters Global TLV

Table I-7. Fields of the Fragmentation Parameters Global TLV

Field Name	Size	Description
Node ID	2	This indicates the node ID of the device that the subsequent fragmentation parameters apply to.
Fragmentation Options	1	This bitfield indicates what fragmentation options are supported by the device. It has the following enumerated bits: Bit 0 = APS Fragmentation Supported. Set to 1 to indicate support; 0 to indicate no support. If set to 1, the maximum reassembled message size is indicated by the Maximum Incoming Transfer Unit. Bit 1-7 = Reserved for future use
Maximum Incoming Transfer Unit	2	This is a copy of the local device's apsMaxSizeASDU AIB value. This indicates the maximum reassembled message size at the application layer after fragmentation has been applied on the message at the lower layers. A device supporting fragmentation would set this field to be larger than the normal payload size of the underlying NWK and MAC layer.

When a device is communicating to other devices it can include this TLV in any message that supports TLVs as a way to convey the size of the largest application layer message it can accept. Earlier versions of the specification do not support TLVs and thus the Node_Desc_rsp is the only means to convey the fragmentation capabilities. By including the short ID in this TLV it allows a router to relay the fragmentation capabilities of the Trust Center since a joiner knows the trust center's address is always 0x0000.

I.4.9 Joiner Encapsulation Global TLV (ID 72)

This TLV can contain one or more TLVs inside of it that are either local or Global. The same rules of TLVs in messages apply to TLVs inside the Encapsulation TLV with the exception that Encapsulation TLVs SHALL NOT contain Encapsulation TLVs.

The data put inside contains information that the joining or rejoining node wants to communicate to the Trust Center.

Octets: Variable
Additional TLVs

Figure I-5. Joiner Encapsulation Global TLV

I.4.10 Beacon Appendix Encapsulation Global TLV (ID 73)

This TLV can contain one or more TLVs inside of it that are either local or Global. The same rules of TLVs in messages apply to TLVs inside the Encapsulation TLV with the exception that Encapsulation TLVs SHALL NOT contain Encapsulation TLVs.

The data put inside contains TLVs that will be part of the Beacon Appendix. Devices will set the contents of their *nwkGlobalBeaconAppendix* NIB value based on the contents of this TLV.

Octets: Variable
Additional TLVs

Figure I-6. Joiner Encapsulation Global TLV

I.4.11 Configuration Parameters Global TLV (ID 75)

The Configuration Parameters Global TLV is 2-bytes in length and indicates various parameters about how the stack SHALL behave. Each bit or bits corresponds to the internal NIB, AIB, or Device Security Policy values and how they are configured. Refer Section 2.4.3.4.4.

Table I-8. Configuration Parameters Global TLV

Bit	Area	Affected Configuration Attribute
0	AIB	apsZdoRestrictedMode
1	Device Security Policy	requireLinkKeyEncryptionForApsTransportKey
2	NIB	nwkLeaveRequestAllowed
3–15	Reserved	Reserved

ANNEX J CRYPTOGRAPHIC PROCESSING FOR ECDHE

This annex contains the cryptographic processing for Elliptic Curve Diffie-Hellman Ephemeral (ECDHE) key negotiation.

J.1 ECDHE/SPEKE Using Curve25519

This section covers SPEKE [B21][B22][B23][B24], a Password Authenticated Key Exchange (PAKE) using Curve25519 [B20] and [B21]. An anonymous (unauthenticated) version can be obtained by employing a well-known password, in which case SPEKE simplifies to ECDHE.

Table J-1. Parameters for ECDHE and SPEKE using Curve25519

Item	Description
i	Initiator, the device initiating the key exchange (generally the joining device)
r	Responder, the device answering the key exchange (generally the Trust Center)
PSK	Pre-shared Key (see Table J-2)
H(x)	Hashing Function. This SHALL be AES-MMO-128 or SHA-256.
H*(x)	Cyclic extension of the hashing function to 256 bits, if the output size of H is less than 256 bit, or a truncated version of H if the output size of H is more than 256 bit. For example, if H(x) = AES-MMO-128, then H*(x) = H(x) || H(x) If the output size is 256-bits, such as SHA-256, a bad generator point avoidance mechanism must be used. This is done by hard coding the first byte to 0x09.
KDF(key, instance)	Key Derivation Function with key, instance as input. This SHALL be either HMAC-AES-MMO-128 (the Specialized Keyed Hash Function for Message Authentication, as defined in section B.1.5), or HMAC-SHA-256-128 (HMAC-SHA-256 truncated to the first 128 bits).
A_i	Initiator Identity, IEEE EUI-64 of Initiator. For purposes of hashing, the IEEE EUI-64 is assumed to be stored in little-endian order.
A_r	Responder Identity, IEEE EUI-64 of Responder. For purposes of hashing, the IEEE EUI-64 is assumed to be stored in little-endian order.
G	Generator (base point)
I	Session Identifier
d_i	Initiator Private Key
d_r	Responder Private Key
Q_i	Initiator Public Key Point where $Q_i = d_iG$ (notice: only the x-coordinate is relevant)
Q_r	Responder Public Key Point where $Q_r = d_rG$ (notice: only the x-coordinate is relevant)
s	Shared Secret
DK	Derived Key

Table J-2. PSK Items

n/a	Anonymous Key Exchange	"Value of *apscWellknownPSK*" (no quotes)
PSK ID 0x00	Key Exchange with Installation Code	Pre-configured link-key derived from Device Installation Code
PSK ID 0x01	Key Exchange with Passcode	Variable-length Passcode, including low-entropy passcodes. Minimum recommended entropy: 20 bits
PSK ID 0x02	Basic Access Key (Zigbee Device)	Details defined in Zigbee Device specification
PSK ID 0x03	Administrative Access Key (Zigbee Device)	Details defined in Zigbee Device specification
PSK ID 0x04 – PSK ID 0x07	Reserved for future use	

J.1.1 Recommendations for Variable-length Passcodes

Variable-length passcodes MAY be used as alternative means of mutual authentication, next to 128-bit pre-configured trust center link-keys derived from installation codes. The purpose of a variable-length passcode is to provide a low entropy shared secret for the class of Password Authenticated Key Exchange (PAKE) protocols, in particular SPEKE over Curve25519. Low-entropy passcodes SHALL NOT be used in connection with protocols that are not designed to support low-entropy passwords, e.g. they are not suitable as pre-shared key for ECDHE-PSK. Low-entropy passcodes offer device manufacturers more flexibility to provide shorter setup codes in order to reduce the amount of data that needs to be typed, spoken or encoded on a barcode displayed on a product. Low-entropy passcodes SHOULD have a minimum entropy of at least 20 bits. This is a compromise between a convenient end-user experience (short codes preferred) and the probability that an active Man-in-the-Middle (MITM) attacker could guess the passcode (longer codes preferred). Given the recommended minimum of 20 bits, the probability of guessing the pre-shared secret would be 1 : 2^20, i.e. less than 0.0001%. Implementations MAY further increase robustness against guessing attacks by adding exponential back-offs for each failed attempt, and by limiting the total number of attempts per device.

J.1.2 Scalar Multiplications on Curve25519

With respect to the Operation stated below, scalar multiplications on the elliptic curve, i.e. Q = dG, SHALL be performed by invoking Q = X25519(d, G) as defined in reference [B20]. The scalar d SHALL first be decoded using d = decodeScalar25519(k), as defined in reference [B20], where k is an octet string of 32 secret random bytes.

J.1.3 Operation

1. Initiator computes G = H*(PSK), generates d_i at random, subject to private key post-processing as detailed below, and computes $Q_i = d_iG$.
2. Initiator sends its Identity A_i and Q_i.
3. Responder, when it receives A_i and Q_i, first checks if another key establishment session with the same Initiator is in progress. If so, it aborts the previous session with failure;it computes G = H*(PSK), generates d_r at random, subject to private key post-processing as detailed below, and computes $Q_r = d_rG$.
4. Responder sends its Identity A_r and Q_r back to initiator.

   The following operations occur in no guaranteed order, potentially concurrently at both ends:

   - Responder computes point $(x_k, y_k) = d_rQ_i$ and proceeds with step 5
   - Initiator generates point $(x_k, y_k) = d_iQ_r$ and proceeds with step 5

5. Initiator/Responder determines the session identifier,

   $$I = \begin{cases} A_r \,||\, Q_r \,||\, A_i \,||\, Q_i, A_r \leq A_i \\ A_i \,||\, Q_i \,||\, A_r \,||\, Q_r, A_r > A_i \end{cases}.$$

   Note: A_r and A_i are compared as 64-bit unsigned integers with the IEEE 802 OUI part being the most significant bits, i.e. the EUI-64 00:AA:00:11:22:33:44:55 is less than 00:BB:00:11:22:33:44:FF. Notice that while these are transferred in little-endian representation, the comparison above is done based on the actual integer values.

6. Initiator/Responder derives shared secret, s = H(x_k || I|| G)
7. Initiator/Responder derives APS link key = KDF(s, 1), which is the outcome of executing the specialized keyed hash function specified in section B.1.5 under the shared secret obtained in step 6 with the 1-octet string ‘0x01’ as the input string.
8. Initiator sends APSME verify key message for the key derived in step 7.
9. Responder sends APSME confirm key message using the key derived in step 7.

J.1.4 Contributory Behavior

There exist points of small order on curve 25519 and its twist, which relate to five unique x-coordinates:

- $x = 0$
- $x = 1$
- $x = -1$
- $x_1 =$ $39382357235489614581723060781553021112529911719440698176882885853963445705823$
- $x_2 =$ $325606250916557431795983626356110631294008115727848805560023387167927233504$

Considering a large integer size of 32 octets, which can represent values up to $2^{256} - 1$, and modulo-p arithmetics with $p = 2^{255} - 19$, the following 12 points result in a (trivial) shared secret $x_k = 0$:

$$x_{0,1} = 0$$

$$x_{0,2} = p \equiv x_{0,1}$$

$$x_{0,3} = 2p \equiv x_{0,1}$$

$$x_{0,4} = 1$$

$$x_{0,5} = p + 1 \equiv x_{0,4}$$

$$x_{0,6} = 2p + 1 \equiv x_{0,4}$$

$$x_{0,7} = p - 1$$

$$x_{0,8} = 2p - 1 \equiv x_{0,7}$$

$$x_{0,9} = x_1$$

$$x_{0,10} = p + x_1 \equiv x_{0,9}$$

$$x_{0,11} = x_2$$

$$x_{0,12} = p + x_2 \equiv x_{0,11}$$

This is also referred to as non-contributory behavior because the private keys of initiator and responder don’t contribute to the shared secret.

As a result the following 32-octet strings should be avoided as generator points (base points for SPEKE):

$x_{0,1}$	0
	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
$x_{0,2}$	57896044618658097711785492504343953926634992332820282019728792003956564819949
	ED FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF 7F
$x_{0,3}$	115792089237316195423570985008687907853269984665640564039457584007913129639898
	DA FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
$x_{0,4}$	1
	01 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
$x_{0,5}$	57896044618658097711785492504343953926634992332820282019728792003956564819950
	EE FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF 7F
$x_{0,6}$	115792089237316195423570985008687907853269984665640564039457584007913129639899
	DB FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
$x_{0,7}$	57896044618658097711785492504343953926634992332820282019728792003956564819948
	EC FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF 7F
$x_{0,8}$	115792089237316195423570985008687907853269984665640564039457584007913129639897
	D9 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
$x_{0,9}$	39382357235489614581723060781553021112529911719440698176882885853963445705823
	5F 9C 95 BC A3 50 8C 24 B1 D0 B1 55 9C 83 EF 5B 04 44 5C C4 58 1C 8E 86 D8 22 4E DD D0 9F 11 57
$x_{0,10}$	97278401854147712293508553285896975039164904052260980196611677857920010525791
	5F 9C 95 BC A3 50 8C 24 B1 D0 B1 55 9C 83 EF 5B 04 44 5C C4 58 1C 8E 86 D8 22 4E DD D0 9F 11 D7
$x_{0,11}$	325606250916557431795983626356110631294008115727848805560023387167927233504
	E0 EB 7A 7C 3B 41 B8 AE 16 56 E3 FA F1 9F C4 6A DA 09 8D EB 9C 32 B1 FD 86 62 05 16 5F 49 B8 00
$x_{0,12}$	58221650869574655143581476130700064557929000448548130825288815391124492053472
	E0 EB 7A 7C 3B 41 B8 AE 16 56 E3 FA F1 9F C4 6A DA 09 8D EB 9C 32 B1 FD 86 62 05 16 5F 49 B8 80

For H(x) = AES-MMO-128, and H*(x) = H(x) || H(x), it is guaranteed that for all possible pre-shared secrets the base point G will always be different than any of the points mentioned above. No special consideration regarding the pre-shared secret, except for its minimum entropy, is required in this case.

For H(x) = SHA-256 it is possible to have a pre-shared secret that will derive generator points of small order as listed previously. To avoid these known, bad generator points this specification defines a simple mechanism to hard code the first byte. This mechanism is simple to implement for embedded devices with limited flash and does not require a re-generation step if a bad point is encountered.

The following defines the mechanism for bad generator point avoidance.

$$H(x) = \text{SHA-256}(x)$$

$$H[0] = \text{0x09}$$

$$H^*(x) = H$$

The following defines the mechanism for bad generator point avoidance.

$$H(x) = SHA\text{-}256(x)$$

$$H[0] = 0x09$$

$$H^*(x) = H$$

ANNEX K ZIGBEE PROVISIONAL AND EXPERIMENTAL FEATURES

This annex describes provisional and experimental features which have been drafted and will be included in future revisions of the specification, but have not undergone any testing for compliance and interoperability validation. Therefore the following features SHALL not be implemented in a platform undergoing formal certification. The purpose for including this text is to ensure implementers consider these features to avoid potential interoperability issues in future specification revisions. These features may include considerations such as ensuring all new and existing frames received are processed regardless of: being TLV extended (unless specified otherwise), supporting reserved TLV Tag Ids, etc.

K.1 Routing Improvements

K.1.1 NLME-ROUTE-DISCOVERY.request

This primitive allows the next higher layer to initiate route discovery.

K.1.1.1 Semantics of the Service Primitive

The semantics of this primitive are as follows:

```
NLME-ROUTE-DISCOVERY.request    {
                                DstAddrMode,
                                DstAddr,
                                Radius,
                                NoRouteCache,
                                SourceRoute
                                }
```

Table K-1 specifies the parameters for this primitive.

Table K-1. NLME-ROUTE-DISCOVERY.request Parameters

Name	Type	Valid Range	Description
DstAddrMode	Integer	0x00 – 0x02	A parameter specifying the kind of destination address provided. The DstAddrMode parameter MAY take one of the following three values: 0x00 = No destination address 0x01 = Reserved 0x02 = 16-bit network address of an individual device
DstAddr	16-bit network address	Any network address	The destination of the route discovery. If the DstAddrMode parameter has a value of 0x00 then no DstAddr will be supplied. This indicates that the route discovery will be a many-to-one discovery with the device issuing the discovery command as a target.

Name	Type	Valid Range	Description
			If the DstAddrMode parameter has a value of 0x02, this indicates unicast route discovery. The DstAddr will be the 16-bit network address of a device to be discovered.
Radius	Integer	0x00 – 0xff	This optional parameter describes the number of hops that the route request will travel through the network.
NoRouteCache	Boolean	TRUE or FALSE	In the case where DstAddrMode has a value of zero, indicating many-to-one route advertisement, this flag determines whether the NWK SHOULD establish a route record table. TRUE = no route record table SHOULD be established FALSE = establish a route record table
SourceRoute	Boolean	TRUE or FALSE	In the case where DstAddrMode has a value of 0x02, indicating unicast route discovery, and the device is operating as a concentrator, this flag determines whether an ad-hoc route is sought, or rather a source route. TRUE = a source route is sought FALSE = a normal ad-hoc route is sought Refer to section 3.6.4.5.1.

K.1.1.2 When Generated

This primitive is generated by the next higher layer of a Zigbee coordinator or router and issued to its NLME to request the initiation of route discovery.

K.1.1.3 Effect on Receipt

On receipt of this primitive by the NLME of a Zigbee end device, the NLME will issue the NLME-ROUTE-DISCOVERY.confirm primitive to the next higher layer with a status value of INV_REQUESTTYPE.

On receipt of this primitive by the NLME with the DstAddrMode parameter not equal to 0x00 and the DstAddr parameter equal to a broadcast address, the NLME will issue the NLME-ROUTE-DISCOVERY.confirm primitive to the next higher layer with a status value of INV_REQUESTTYPE.

On receipt of this primitive by a Zigbee router or Zigbee coordinator that has routing capacity, with the DstAddrMode parameter equal to 0x02, the NLME will initiate discovery of a unicast route between the current device and the network device with the 16-bit network address given by the DstAddr parameter. The procedure for initiating discovery of a unicast route is outlined in section 3.6.4.5.1.

On receipt of this primitive on a Zigbee router or Zigbee coordinator with concentrator capabilities, if the DstAddrMode parameter equal to 0x00, the NLME will initiate many-to-one route advertisement. The procedure for initiating many-to-one route advertisement is outlined in section 3.6.4.5.1.

On receipt of this primitive on a Zigbee router or Zigbee coordinator with concentrator capabilities, if the DstAddrMode parameter equal to 0x02 and the SourceRoute flag equal to TRUE, the NLME will initiate many-to-one route advertisement with an appended Source Route Solicitation TLV, which includes DstAddr in the nwkAddressOfInterest list. The procedure for initiating many-to-one route advertisement is outlined in section 3.6.4.5.1.In each of the three cases of actual route discovery described above, the NLME will initiate route discovery by attempting to transmit a route discovery command frame using the MCPS-DATA.request primitive of the MAC sub-layer. If a value has been supplied for the optional Radius parameter, that value will be placed in the Radius field of the NWK header of the outgoing frame. If a value has not been supplied then the radius field of the NWK header will be set to twice the value of the *nwkcMaxDepth* attribute of the NIB as would be the case for data frame transmissions. If the MAC sub-layer fails, for any reason, to transmit the route request command frame, the NLME will issue the NLME-ROUTE-DISCOVERY.confirm primitive to the next higher layer with a Status parameter value equal to that returned by the MCPS-DATA.confirm. If the route discovery command frame is sent successfully and if the DstAddrMode parameter has a value of 0x00, indicating many-to-one route advertisement, the NLME will immediately issue the NLME-ROUTE-DISCOVERY.confirm primitive with a value of SUCCESS. Otherwise, the NLME will wait until either a route reply or route record command frame is received or a reactive many-to-one route request command originating in the device identified by DstAddr is received or the route discovery operation times out as described in section 3.6.4.5. If a route reply or route record or matching reactive many-to-one route request command frame is received before the route discovery operation times out, the NLME will issue the NLME-ROUTE-DISCOVERY.confirm primitive to the next higher layer with a status value of SUCCESS. If the operation times out, it will issue the NLME_ROUTE-DISCOVERY.confirm primitive with a Status of ROUTE_ERROR and with a NetworkStatusCode value reflecting the reason for failure as described in Table 3-52.

K.2 Route Request Command

K.2.1 TLVs

An optional list of tag-length-value records with context-specific information, which is relevant to the Route Request. The following TLVs MAY be appended to the Route Request Command and when present SHALL be parsed and processed by the recipient in context. The context is established by the values of other relevant fields within the same Network Command frame, i.e. fields in the basic command frame and potentially leading TLVs.

K.2.1.1 Tag ID 0: Extended Route Information

The Extended Route Information TLV conveys additional route-specific information. It SHALL be included in all Route Request Commands. Notice that devices built to earlier revisions of this specification did not support this TLV and implementations SHALL tolerate the TLV being absent.

Octets: 2	**1**
Routing Sequence Number	Initial Radius

Figure K-1. Extended Route Information TLV Payload Format

K.2.1.1.1 Routing Sequence Number Field

The Routing Sequence Number Field SHALL contain the value of nwkRoutingSequenceNumber after it has been incremented by one. This field denotes the Routing Sequence Number of the originator of the frame.

K.2.1.1.2 Initial Radius Field

The Initial Radius Field SHALL contain the value of the Radius field in the NWK Header of the original route request, i.e. the first instance of any particular route request frame.

K.2.1.2 Tag ID 1: Concentrator Information

The Concentrator Information TLV conveys the settings of the concentrator device, which initiated the Route Request. It SHALL be included when the Destination Address equals 0xfffc (All Routers and the Coordinator). Notice that devices built to earlier revisions of this specification did not support this TLV and implementations SHALL tolerate the TLV being absent even if the Destination Address equals 0xfffc. A device which has *nwkIsConcentrator* set as TRUE, and which originates a many-to-one route request, MUST populate this TLV prior to broadcasting the route request frame. Devices forwarding the route request MUST forward this TLV without modification.

1	**1**
Concentrator Discovery Time	Max Source Route Length

Figure K-2. Concentrator Information TLV Payload Format

K.2.1.2.1 Concentrator Discovery Time Field

The Concentrator Radius Field SHALL contain the value of nwkConcentratorDiscoveryTime.

K.2.1.2.2 Concentrator Max Source Route Length Field

The Concentrator Radius Field SHALL contain the value of nwkMaxSourceRoute.

K.2.1.3 Tag ID 2: Source Route Solicitation

The Source Route Solicitation TLV conveys a list of one or more network short addresses of devices the concentrator is interested in establishing a source route to. It MAY be included when the Destination Address equals 0xfffc (All Routers and the Coordinator).

Octets: 2	**2/0**	**…**	**2/0**
nwkAddressOfInterest1	nwkAddressOfInterest 2	…	nwkAddressOfInterest n

Figure K 3. Source Route Solicitation TLV Payload Format

K.2.1.3.1 nwkAddressOfInterest Fields

Each nwkAddressOfInterest field SHALL contain the value the network short address of a destination device which the requesting device wants to establish a source route to.

K.3 Route Reply Command

K.3.1 Local TLVs

An optional list of tag-length-value records with context-specific information, which is relevant to the Route Reply. The following TLVs MAY be appended to the Route Reply Command and when present SHALL be parsed and processed by the recipient in context. The context is established by the values of other relevant fields within the same Network Command frame, i.e. fields in the basic command frame and potentially leading TLVs.

K.3.1.1 Tag ID 0: Extended Route Information

The Extended Route Information TLV conveys additional route-specific information. It SHALL be included in all Route Reply Commands. Notice that devices built to earlier revisions of this specification did not support this TLV and implementations SHALL tolerate the TLV being absent.

Octets: 2
Routing Sequence Number

Figure K-4. Extended Route Information TLV Payload Format

K.3.1.1.1 Routing Sequence Number Field

The Routing Sequence Number Field SHALL contain the value of nwkRoutingSequenceNumber after it has been incremented by one. This field denotes the Routing Sequence Number of the Responder.

K.4 Network Status Command

K.4.1 TLVs

An optional list of tag-length-value records with context-specific information, which is relevant to the Network Status command. The following TLVs MAY be appended to the Network Status Command and when present SHALL be parsed and processed by the recipient in context. The context is established by the values of other relevant fields within the same Network Command frame, i.e. fields in the basic command frame and potentially leading TLVs.

K.4.1.1 Tag ID 0: Extended Route Information

The Extended Route Information TLV conveys additional route-specific information. It SHALL be included in those Network Status commands, which convey route errors, specifically the Link Failure and Many-To-One Route Failure; it is not required for source route failures and statuses not related to routing. Notice that devices built to earlier revisions of this specification did not support this TLV and implementations SHALL tolerate the TLV being absent. If the related routing table entry, in its sequence number valid flag, indicates that the sequence number is unknown or invalid, the extended route information TLV SHALL NOT be included.

Octets: 2
Routing Sequence Number

Figure K-5. Extended Route Information TLV Payload Format

K.4.1.1.1 Routing Sequence Number Field

The Routing Sequence Number Field SHALL contain the value of sequence number field of the routing table entry that was utilized to forward a frame to its next hop and such attempt failed due to a link failure. This applies, both, to link failures along ad-hoc routes, as well as link failures along many-to-one routes.

K.5 Initiation of Route Discovery

If the device initiating route discovery is currently operating as a concentrator, as indicated by the nwkIsConcen-trator flag, and has not been specifically instructed by the NHLE to seek a normal ad-hoc route versus a source route, it SHOULD prefer discovery of source routes over discovery of ad-hoc routes. It still MAY perform normal ad-hoc route discovery, e.g. to avoid the per-frame source route overhead, unless specifically instructed to seek a source route by the SourceRoute parameter of the NLME-ROUTE-DISCOVERY.request.

If the device initiating route discovery has no routing table entry corresponding to the routing address of the destination device, and intends to perform a normal ad-hoc route discovery, it SHALL establish a routing table entry with status equal to DISCOVERY_UNDERWAY. If the device has an existing routing table entry corresponding to the routing address of the destination with status equal to ACTIVE, that entry SHALL be used and the status field of that entry SHALL retain its current value. If it has an existing routing table entry with a status value other than ACTIVE, that entry SHALL be used and the status of that entry SHALL be set to DISCOVERY_UNDERWAY. The device SHALL also establish the corresponding route discovery table entry if one with the same initiator and route request ID does not already exist.

Each device issuing a route request command frame SHALL maintain a counter used to generate route request identifiers. When a new route request command frame is created, the route request counter is incremented and the value is stored in the device's route discovery table in the Route request identifier field. The device SHALL incre-ment nwkRoutingSequenceNumber and append an Extended Route Information TLV to the route request com-mand frame with the Route Sequence Number field set to the resulting value. Other fields in the routing table and route discovery table are set as described in section 3.6.4.2.

Each device issuing a route request command frame SHALL maintain a counter used to generate route request identifiers. When a new route request command frame is created, the route request counter is incremented and the value is stored in the device's route discovery table in the Route request identifier field. The device SHALL increment *nwkRoutingSequenceNumber* and append an Extended Route Information TLV to the route request command frame with the Route Sequence Number field set to the resulting value. Other fields in the routing table and route discovery table are set as described in section 3.6.4.2.

The many-to-one route advertisement procedure SHALL be initiated by the NWK layer of a Zigbee router or coordinator on receipt of an NLME-ROUTE-DISCOVERY.request primitive from the next higher layer where the DstAddrMode parameter has a value of 0x00, or where the DstAddrMode parameter has a value of 0x02 and the SourceRoute parameter is set as TRUE. A many-to-one route request command frame is not retried; however, a discovery table entry is still created to provide loop detection during the *nwkcRouteDiscoveryTime* period. If the NoRouteCache parameter of the NLME-ROUTE-DISCOVERY.request primitive is TRUE, the many-to-one sub-field of the command options field of the command frame payload SHALL be set to 2. Otherwise, the many-to-one sub-field SHALL be set to 1. Note that in this case, the NWK layer should maintain a route record table. The destination address field of the NWK header SHALL be set to 0xfffc, the all-router broadcast address. The broadcast radius SHALL be set to the value in *nwkConcentratorRadius*. A source device that initiates a many-to-one route advertisement is designated as a concentrator and referred to as such in this document and the NIB attribute *nwkIsConcentrator* should be set to TRUE. If a device has *nwkIsConcentrator* equal to TRUE and there is a non-zero value in *nwkConcentratorDiscoveryTime*, the network layer should issue a route request command frame each *nwkConcentratorDiscoveryTime*, making sure that any two consecutive many-to-one route request commands with different route request identifier are separated in time by at least *nwkConcentratorDiscoverySeparation*.

If the DstAddrMode parameter has a value of 0x02 and the SourceRoute parameter is set as TRUE, the device SHALL include the Source Route Solicitation TLV in the route request command frame, and include DstAddr in the nwkAddressOfInterest list in that TLV. In order to improve interoperability with legacy routers built to earlier revisions of this specification, the device MAY additionally instigate an IEEE_addr_req with the NWKAddressOfInterest parameter set as DstAddr and RequestType either set as 0 – 'Single device response' or 1 – 'Extended response'; the latter allows the router to learn about source routes to end-devices in a single step in case the device denoted by DstAddr is also a router. This additional IEEE_addr_req SHALL be subject to the same throttling provided by *nwkConcentratorDiscoverySeparation*.Upon Receipt of a Route Request Command Frame.

K.5.1 Initiating a Route Reply or Reactive Many-to-One Route Request

The device SHALL check if it is the intended destination. If it is the intended destination and the device is currently operating as a concentrator, it SHALL determine whether the originator of the route request is within the concentrator radius by calculating the distance to the originating router according to section 3.6.4.5.5; if the calculated distance is less than or equal to the concentrator radius, it SHALL issue a reactive many-to-one route request, instead of responding with a route reply command frame making sure that any two consecutive many-to-one route request commands with different route request identifiers are separated in time by at least nwkConcentratorDiscoverySeparation. It SHALL also check if the destination of the command frame is one of its end device children by comparing the destination address field of the route request command frame payload with the address of each of its end device children, if any. If either the device or one of its end device children is the destination of the route request command frame, and it is not issuing a reactive many-to-one route request, it SHALL reply with a route reply command frame.

When replying to a route request with a route reply command frame, the device SHALL construct a frame with the frame type field set to 0x01. The route reply's source address SHALL be set to the 16-bit network address of the device creating the route reply and the destination address SHALL be set to the calculated next hop address, considering the originator of the route request as the destination. The link cost from the next hop device to the current device shall be computed as described in section 3.6.4.1 and inserted into the path cost field of the route reply command frame. The device SHALL increment *nwkRoutingSequenceNumber* and the resulting value is appended to the route reply command frame using the Routing Sequence Number TLV The route reply command frame SHALL be unicast to the next hop device by issuing an MCPS-DATA.request primitive.

K.5.2 Routing and Route Discovery Table Maintenance, Route Request Forwarding

When a device with routing capacity is not the destination of the received route request command frame, it SHALL determine if a route discovery table entry (see Table 3-75) exists with the same route request identifier and source address field. If no such entry exists, one SHALL be created. The route request timer SHALL be set to expire in *nwkcRouteDiscoveryTime* OctetDurations. If a routing table entry corresponding to the routing address of the destination exists and its status is not ACTIVE, the status SHALL be set to DISCOVERY_UNDERWAY. If no such entry exists and the frame is a unicast route request, an entry SHALL be created and its status set to DISCOVERY_UNDERWAY. If the frame is a many-to-one route request, the device SHALL also create a routing table entry with the destination address field equal to the source address of the route request command frame by setting the next hop field to the address of the previous device that transmitted the command frame. If the frame is a many-to-one route request (*i.e.* the many-to-one sub-field of the command options field of the command frame payload has a non-zero value), the many-to-one field in the routing table entry SHALL be set to TRUE, the route record required field SHALL be set to TRUE if the coordinator is within source route range or FALSE otherwise, and the no route cache flag SHALL be set to TRUE if the many-to-one sub-field of the command options field of the command frame payload has a value of 2 or to FALSE if it has a value of 1. If the routing table entry is new, or if the no route cache flag is set to TRUE, or if the next hop field changed, the route record required field SHALL be set to TRUE, if the coordinator is within source route range otherwise it remains unchanged. The coordinator is deemed within source route range, if the distance as calculated section 3.6.4.5.4 yields a value less than or equal to the Max Source Route Length field in the Concentrator Information TLV of the received route request command frame; for backwards compatibility with legacy implementations, it is also deemed within source route range if no Concentrator Information TLV is present at all. The status field SHALL be set to ACTIVE.

K.5.3 Transmitting Reactive Many-To-One Route Requests

If a concentrator determines that the route request has originated from a router, which is operating within its *nwkConcentratorRadius*, as calculated according to section 3.6.4.5.5, it SHALL, instead of sending a route reply, initiate route discovery as outlined in section 3.6.4.5.1 by instigating an NLME-ROUTE-DISCOVERY.request primitive with the DstAddr set to the source address field of the network header of the route request, the DstAddrMode parameter set as 0x02, and the SourceRoute parameter set as TRUE.

K.5.4 Calculating the Distance Between Routers and Concentrators

A concentrator can determine the distance to a router in the network by subtracting the value conveyed in the radius field of the network header of a route request command, which it received from that router, from either (i) the Initial Radius field of the Extended Route Information TLV, or, (ii) twice the value of the nwkcMaxDepth NIB attribute, if such a TLV was not included in the route request command frame. The result of this calculation is the number of hops, minus one, a packet travels along the current route between router and concentrator. A message sent from the concentrator with the initial radius set to the calculated distance can be assumed to reach the router device unless the network topology changed.

K.5.5 Initiation and Processing of a Route Record Command Frame

If the NWK layer of a Zigbee router or Zigbee coordinator is forwarding a unicast data frame on behalf of one of its end device children and the many-to-one field of the destination's routing table entry has a value of TRUE, then the device SHALL unicast a route record command to the destination before relaying the data frame, which al-ready contains the network short address of the Zigbee router or Zigbee coordinator in the relay list.

If the NWK layer of a Zigbee router or Zigbee coordinator receives a many-to-one route request, which includes the Source Route Solicitation TLV and the receiving device's network short address appears in the nwkAddressOfInterest list of that TLV and the receiving device is within source route range it SHALL set the route record required flag in the corresponding many-to-one routing table entry; it SHALL then also unicast a route record command to the originator of the many-to-one route request, with the relay list being empty; similarly, if the network short address of any of its end device children appears in nwkAddressOfInterest and the receiving device is within source route range minus one (accounting for the additional hop to the end device child), the NWK SHALL set the route record re-quired flag in the corresponding many-to-one routing table entry and then also unicast a route record command to the originator of the many-to-one route request for each matching network short address, with the relay list already containing the network short address of the Zigbee router or Zigbee coordinator.

The coordinator is deemed within source route range, if the distance as calculated section 3.6.4.5.5 yields a value less than or equal to the Max Source Route Length field in the Concentrator Information TLV of the received route request command frame; for backwards compatibility with legacy implementations, it is also deemed within source route range if no Concentrator Information TLV is present at all.

An optional optimization is possible in which the router or coordinator MAY keep track of which of its end device children have received source routed data frames from a particular concentrator device and can thereby reduce the number of route record commands it transmits to that concentrator on behalf of its end device children.

