Zigbee PRO Green Power feature specification Basic functionality set

Version 1.1.1

Zigbee Document 14-0563-18

January 11th, 2019

Sponsored by: Zigbee Alliance

Accepted by	This document has been accepted for release by the ZigBee Alliance Board of Directors
Abstract	This document is a maintenance release of the Green Power Basic v1.1 specification, containing all applicable errata.
Keywords	Zigbee, Green Power, Battery-less, Energy Harvesting, Green Power stub, Green Power Cluster, Green Power Basic, generic switch, Compact Attribute Reporting, multi-sensor, set-point

This page is intentionally blank

Notice of use and disclosure

This page is intentionally blank

38
Revision history

Revision	Version	Date	Details	Editor
000	0.0	July 7th, 2016	Baseline: clean Green Power Basic specification (14-0563-016)	Bozena Erdmann
001	0.0	July 20th, 2016	First draft, based on the multi-sensor baseline (16-02605)	Bozena Erdmann
002	0.0	August 3rd, 2016	Implementing comments as discussed during Green Power WG call on July 27th and received via email from Jorgen van Parys on July 28th	Bozena Erdmann
003	0.0	August 9th, 2016	Implementing the missing Translation Table representation of the multi-sensor data, according to the baseline (16-02605); Adding editorial and technical comments for discussion during August 10th GP WG call	Bozena Erdmann
004	0.0	August 10th, 2016	Documenting resolutions to technical comments as agreed to during GP WG call of August 10	Bozena Erdmann
005	0.0	August 13th, 2016	Implementing the resolutions as agreed to during GP WG call of August 10; for ease of review, all new spec elements are indicated by a comment with the text "new in r005"	Bozena Erdmann
006	0.0	August 17th, 2016	Implementing the resolution as agreed to during GP WG call of August 17: making the "Number of report" fields of the GPD Application description always present; the changes are indicated by a comment with the text "new in r006"	Bozena Erdmann
007	0.0	August 24nd, 2016	Fixing editorials reported by Arasch Honarbacht via email; implementing the resolutions as agreed during the GP WG call of August 24; documenting further open questions. The changes are indicated by a comment with the text "New in r007"	Bozena Erdmann
008	0.0	August 24nd, 2016	Additional clarification on the usage of the *Number of attribute records* field of the GPD Application Description command The changes are indicated by a comment with the text "New in r007"	Bozena Erdmann
009	0.0	August 24nd, 2016	Additional clarification on the usage of the *Remaining attribute record Length* field of the GPD Application Description command The changes are indicated by a comment with the text "New in r007"	Bozena Erdmann
010	0.7 candidate	September 9th, 2016	Implementing comments from the GP multi-sensor August PoC, Zigbee document 16-02611	Bozena Erdmann
011	0.7 candidate	September 14th, 2016	Fixing editorial comments received via email.	Bozena Erdmann
012	0.7 candidate	October 4th, 2016	Implementing resolutions for the comments from the GP multi-sensor v0.7 letter ballot	Bozena Erdmann
013	0.7 candidate	October 11th, 2016	Implementing resolutions for the comments from the GP multi-sensor v0.7 letter ballot: comment #783, #782	Bozena Erdmann
014	0.7 candidate	October 12th, 2016	Implementing resolutions for the comments from the GP multi-sensor v0.7 letter ballot: comment #773	Bozena Erdmann
015	0.7	October 22nd, 2016	Merging the GP multi-sensor v0.7 specification with the GP generic switch v0.7 specification (16-02013-009)	Bozena Erdmann
016	0.7	November 3rd, 2016	Including commissioning guidelines for multiple buttons of the GPD generic switch (16-02604-004)	Bozena Erdmann
017	0.7	November 14th, 2016	Implementing comments from the GP October multi-sensor and generic switch PoC: #960 - #965.	Bozena Erdmann
018	0.7	November 18th, 2016	Adding a clarification for a special case of Translation Table entry with Additional Information for GPD 8-bit vector: press command with *Contact bitmask* = 0x00	Bozena Erdmann
019	0.9 candidate	December 3rd, 2016	Implementing resolutions to GP multi-sensor LB v0.9 comments: #973, #972, #975.	Bozena Erdmann
020	0.9	February 11th, 2017	Implementing resolutions to comments from GP generic switch and multi-sensor December '16 SVE: #1012, #1013, #1014, #1024, #1026, #1027.	Bozena Erdmann

021	0.9	February 13th, 2017	Implementing resolutions to comments from GP generic switch and multi-sensor December '16 SVE: #1029. Implementing comments from the v0.9 TSC review: #1043, #1044.	Bozena Erdmann
022	1.0 candidate	June 15th, 2017	Implementing resolutions to comments from GP generic switch and multi-sensor May '17 SVE. Integrated approved GP Basic errata from 15-02014r008.	Bozena Erdmann
023	1.0 candidate	January 24th, 2018	**Integrated draft GP Basic errata from 15-02014r014:** CCBs: #2336, #2337, #2417, #2418, #2419, #2416, #2415, #2431, #2420, #2408, #2397, #2394, #2380, #2323, #2325, #2326, #2344, #2345, #2346, #2353, #2360, #2361, #2362, #2372, #2375, #2378, #2379, #2421, #2422, #2424, CCB #2447 as described in 17-02671-004. Implementing comment resolution from letter ballot for GP Basic errata set: Kavi comment #1378. Implementing resolution for CCB #2470, #2471, #2517.	Bozena Erdmann
024	1.0 candidate	February 21st, 2018	Implementing resolution for CCB #2528.	Bozena Erdmann
025	1.0 candidate	November 7th, 2018	Implementing resolutions for CCB #2743, #2626, #2565, #2570; #2719; #2736	Bozena Erdmann
026	1.0 candidate	November 7th, 2018	**Implementing comments from the November vScVE:** Kavi comment #2107, #2106.	Bozena Erdmann
027	1.0 candidate	December 16th, 2018	**Implementing comments from the December superballot:** Kavi comment #2174, #2189.	Bozena Erdmann

Table of Contents

Contents

This page is intentionally blank

181 # [1]List of Figures

[1] v0.9 TSC approval comment #1044: https://workspace.zigbee.org/kws/groups/PRO_GP/comments/view_comment?comment_id=1044

184 # [2]List of Table

[2] v0.9 TSC approval comment #1044: https://workspace.zigbee.org/kws/groups/PRO_GP/comments/view_comment?comment_id=1044

1 Introduction

1.1 Scope

This document describes all the technical aspects related with the Green Power feature, incl. the specification of the Green Power Device definitions and frame format, Green Power Proxy and Green Power Sink definitions, and behavior, incl. Green Power cluster specification, Green Power stub specification, and commissioning procedures.

1.2 Purpose of the Document

This document contains the specification of the Green Power feature.

2 References

2.1 Normative references

2.1.1 Zigbee Alliance documents

[1] Zigbee document 053474r21 (or later release), Zigbee Specification

[2] Zigbee document 08006, Zigbee-2007 Layer PICS and Stack Profiles

[3] Zigbee document 075123r06, Zigbee Cluster Library Specification r06

[4] Zigbee document 094991, Green Power Technical Requirements Document (TRD)

[5] Zigbee document 15-0015r14, Green Power Basic test specification v1.1.1

[6] Zigbee document 15-0006r13, Green Power Basic PICS v1.1.1

[7] Zigbee document 053874, Zigbee Manufacturer Code Database

[8] Zigbee document 106138, Recommendation for Zigbee PRO Interoperability Across Profiles

[9] Zigbee document 115337, Green Power SrcID Policy Proposal

[10] Zigbee document 106050r03, Zigbee Device Interworking

[11] Zigbee document 115456r04 or later, Master Cluster List

[12] Zigbee document 120525, Product Details Guidelines

[13] Zigbee document 13-0166r01, Master List of Green Power Device Definitions

[14] Zigbee document 15-02014, Errata for Green Power Basic specification

[15] Zigbee document 13-0589, Zigbee Application Architecture, revision 13 or later

2.1.2 ISO / IEEE Standards Documents

[16] [3]Institute of Electrical and Electronics Engineers, Inc., IEEE Std. 802.15.4 2011, IEEE Standard for Information Technology Telecommunications and Information Exchange between Systems – Local and Metropolitan Area Networks – Specific Requirements Part 15.4: Wireless Medium Access Control (MAC) and Physical Layer (PHY) Specifications for Low Rate Wireless Personal Area Networks (WPANs). New York: IEEE Press. 2011

[17] 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/.

2.2 Informative references

2.2.1 Zigbee Alliance documents

[18] Zigbee document 053520, Zigbee Home Automation Profile Specification

[19] Zigbee document 105859, Zigbee Building Automation Profile Specification

[3] https://workspace.zigbee.org/higherlogic/ws/groups/PRO_GP/comments/view_comment?comment_id=1048

[20] Zigbee document 11197, GP best practices for ZHA

[21] Zigbee document 11196, GP best practices for ZBA

3 Definitions

3.1 Conformance levels

Expected: A key word used to describe the behavior of the hardware or software in the design models assumed by this profile. Other hardware and software design models MAY also be implemented.

MAY: A key word indicating a course of action permissible within the limits of the standard (MAY equals is permitted).

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

SHOULD**:** A key word indicating that, among several possibilities, one is recommended as particularly suitable, without mentioning or excluding others; that a certain course of action is preferred but not necessarily required; or, that (in the negative form) a certain course of action is deprecated but not prohibited (SHOULD equals is recommended that).

3.2 Conventions

3.2.1 Number formats

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

3.2.2 Transmission order

The frames in this specification are described as a sequence of fields in a specific order. All frame formats are depicted in the order in which they are transmitted by the PHY, 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 MAC in the order from the octet containing the lowest numbered bits to the octet containing the highest numbered bits.

3.2.3 [4]Reserved values

To support [5]backward- and forward-compatibility, devices SHALL ignore any values or bit settings for any reserved field or sub-field, and SHALL try to process the frame. If the field or sub-fields is necessary for interpreting or necessary for use in conjunction with other fields, the whole message MAY be ignored.
The future definition of the fields and sub-fields reserved in the current version of the specification, unless explicitly stated otherwise, is reserved solely for Zigbee specifications; manufacturers SHALL NOT use the reserved sub-field or reserved field values or bit settings.

Unless explicitly specified otherwise, devices SHOULD try to process a frame with a defined field or sub-field set to a value which is marked as a reserved value according to the specification the device is implemented against. Devices SHALL NOT try to process a frame with ClusterIDs, and cluster-specific CommandIDs and AttributeIDs which they do not support; the ZCL [3] specifies rules for reporting error in such a case.
The future definition of the reserved values of fields and sub-fields, unless explicitly stated otherwise,

[4] CCB #2325; Resolution added in 15-02014-011
[5] LB v07: https://workspace.zigbee.org/kws/groups/zigbee_pro_foundation/comments/view_comment?comment_id=317

is reserved solely for Zigbee specifications; manufacturers SHALL NOT use the reserved values of sub-fields or fields.

To enable future growth and ensure [6]backward- and forward-compatibility, any existing devices which encounter any fields applied after the end of a command SHALL treat them as reserved fields.

The future addition of fields applied after the end of defined cluster commands are reserved solely for Zigbee specifications; Manufacturers SHALL NOT add fields after the end of commands.

3.3 Zigbee Definitions

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

Cluster: A collection of related attributes and commands, which together define a communications interface between two devices. The devices implement server and client sides of the interface respectively.

Cluster identifier: A 16-bit number unique within the scope of an application profile which identifies a specific cluster.

Device: A device consists of one or more Zigbee device descriptions and their corresponding application profile(s), each on a separate endpoint, that share a single 802.15.4 radio (see [16]). Each device has a unique 64-bit IEEE address.

Device Description: A collection of clusters and associated functionality implemented on a Zigbee endpoint. Device descriptions are defined in the scope of an application profile. Each device description has a unique identifier that is exchanged as part of the discovery process.

Node: Same as a device.

Product: A product is a unit that is intended to be marketed. It MAY implement a combination of private, published, and standard application profiles.

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 (see [1]).

Zigbee Coordinator: An IEEE 802.15.4-2003 PAN coordinator (see [16]).

Zigbee End Device: An IEEE 802.15.4-2003 RFD (Reduced Function Device) or FFD (Full Function Device) (see [16]) participating in a Zigbee network, which is neither the Zigbee coordinator nor a Zigbee router.

Zigbee Router: An IEEE 802.15.4-2003 FFD (Full Function Device) participating in a Zigbee network, which is not the Zigbee coordinator but MAY act as an IEEE 802.15.4-2003 coordinator within its personal operating space, that is capable of routing messages between devices and supporting associations.

3.4 Definitions specific to Green Power feature

Application endpoint – Any endpoint other than the dedicated Green Power End Point, hosting application control functionality.

(In)active (Proxy Table) entry – Proxy Table entry, for which the EntryActive flag is set to TRUE (FALSE), respectively.

[6] LB v07: https://workspace.zigbee.org/kws/groups/zigbee_pro_foundation/comments/view_comment?comment_id=317

(In)valid (Proxy Table) entry – Proxy Table entry, for which the EntryValid flag is set to TRUE (FALSE), respectively.

Broadcast – Whenever NWK level broadcast transmission is mentioned within this specification for the GP-defined commands without further description, or where no further description is provided by the Zigbee specification for the Zigbee-defined commands, the RxOnWhenIdle=TRUE (0xfffd) broadcast address SHALL be used.

Direct mode – Sink receiving directly the GPFS in GPD frame format sent by GPD, if in the radio range of the GPD.

(GPD command) Execution (at the sink) - all actions of the GP endpoint of the GP sink leading to providing GP application input to the application on the same radio node. The actions may include GPD command translation, mapping to local application endpoints, forwarding to local application endpoints, local GPD storage, update of attributes, combination with other control inputs, and user feedback.

Fully Compliant Zigbee Device – Device implemented according to Zigbee 2007 or Zigbee PRO stack profile, having the role of either ZR or ZED.

Green Power Device Frame (GPDF) – Special frame format according to the Green Power specification, which is transmitted by or received by GPD.

Groupcast – One of the communication modes used for tunneling GPD commands between the proxies and sinks. In Zigbee terms, it is the APS level multicast, with NWK level broadcast to the RxOnWhenIdle=TRUE (0xfffd) broadcast address.

Pairing – The unidirectional logical link between a Green Power Device and a destination endpoint, which MAY exist on one or more sinks, which makes the sink handle the commands received from this particular GPD. Of particular importance is the configuration procedure leading to the establishment of this special relationship.

Portability – Ability to re-establish communication at a different location, without interruption or re-commissioning.

Green Power End Point (GPEP) – a dedicated reserved endpoint, residing on top of the GP stub, hosting the Green Power cluster.

Tunneled mode – Sink receiving the GPFS forwarded by a proxy located in the radio range of the GPD. This forwarding uses a normal Zigbee frame format but a specific ZCL command from the Green Power cluster: the GP Notification command. [7]The exact conditions for sending the GP Notification command are determined by the *CommunicationMode* sub-field of the Proxy Table entry, defining two groupcast and two unicast modes, see Table 27.

Data GPDF – Any GPDF that carries a GPD Command other than GPD Commissioning (0xE0) or GPD Commissioning Reply (0xF0) or GPD Decommissioning (0xE1), GPD Channel Request (0xE3), GPD Channel Configuration (0xF3), [8]GPD Application Description (0xE4) or any other GPD command from the GPD CommandID range 0xE0 – 0xEF.

GPD Data command – Any GPD Command other than GPD Commissioning (0xE0) or GPD Commissioning Reply (0xF0), GPD Decommissioning (0xE1), GPD Success (0xE2), GPD Channel Request (0xE3), GPD Channel Configuration (0xF3), [9]GPD Application Description (0xE4) or any other GPD command from the GPD CommandID range 0xE0 – 0xEF.

[7] CCB #2276: Resolution added in 15-02014-006
[8] Dec 2016 SVE comment: https://workspace.zigbee.org/kws/groups/PRO_GP/comments/view_comment?comment_id=1026
[9] Dec 2016 SVE comment: https://workspace.zigbee.org/kws/groups/PRO_GP/comments/view_comment?comment_id=1026

Green Power Device (GPD) – A self-powering, energy-harvesting device that implements the Green Power feature.

Green Power Device (GPD) ID – Unique identifier of the GPD, either the 4B SrcID or the IEEE address.

Green Power Proxy Basic (GPPB) or **Basic Proxy** – A proxy that only implements the basic GP proxy functionality, as defined in section 0.

Green Power Manager (GPM) - A Zigbee device capable of managing Green Power functionality, during commissioning or operation, e.g. a GP Commissioning Tool.

Green Power Proxy (GPP) or **proxy** – A fully compliant Zigbee device, which in addition to the core Zigbee specification also implements proxy functionality of the Green Power feature. The proxy is able to handle GPDFs and acts as an intermediate node between the GPD and sinks on the Zigbee network.

Green Power Sink (**GPS**) or **sink** – A fully compliant Zigbee device, which in addition to a core Zigbee specification also implements the sink functionality of the Green Power feature, basic or advanced. The sink is thus capable of receiving, processing and executing GPD commands, tunneled and optionally also directly received.

Green Power Target (GPT) or **Target** – A fully compliant Zigbee device, which in addition to a core Zigbee specification also implements the sink functionality of Green Power Cluster, allowing for receiving, processing and executing tunneled GPD commands, as defined in section A.3.2.1. In the current version of the specification, a GPT can only be implemented on a ZED, because implementation of Basic Proxy is mandatory for ZR.

Green Power Target+ (GPT+) or **Target+** – A Target which also implements the GP stub. A Target+ can thus receive, process and execute both tunneled and directly received GPD commands, as defined in section A.3.2.2. In the current version of the specification, a GPT can only be implemented on a ZED, because implementation of Basic Proxy is mandatory for ZR.

Green Power Combo (GPC) or **Combo** – A fully compliant Zigbee device, which in addition to a core Zigbee specification also implements both the proxy and the sink functionality of the Green Power feature. A Combo can thus receive, process and execute both tunneled and directly received GPD commands (in its sink role), as well as forward them to other GP nodes (in its proxy role).

Green Power Combo Basic (GPCB) or **Basic Combo** – A combo that only implements the basic GP combo functionality, for both sink and proxy, as defined in section A.3.2.7.

Common Green Power Stub (cGP) – Term used for describing the common functionality of Green Power for sending and receiving data packets.

Dedicated Green Power Stub (dGP) – Term used for describing the dedicated Green Power application.

Dedicated LPED Stub (dLPED) – Term used for describing the dedicated Low Power End Device Application (defined by the Low Power End Device task group).

Maintained switch – a switch that stays in its active position state until actuated into a new one, and then remains in that state until acted upon once again.

Momentary switch - a switch that only remain in its active position as long as it is actuated (pressed, held, magnetized, etc.). If not being actuated, it remains in its neutral position.

Rocker, rocker switch – a switch that can be actuated in one of two ways at a time, typically by tapping or pressing on top or bottom part, whereby the switch [10]mechanical design physically prevents both types of actuation at the same time. In case of a realization using the GPD 8-bit vector press

[10] LB v07: https://workspace.zigbee.org/kws/groups/zigbee_pro_foundation/comments/view_comment?comment_id=280

command, both types of actuation result in a different vector (contact status). A Green Power rocker switch is typically a momentary switch. Implementing a Green Power rocker switch as a maintained switch may also be possible; however, such a switch will send two commands on each action (release of the previous action and press of the new action), which can happen to arrive at the receiving application in reversed order; that should then be taken into account in the application.

Pushbutton, button, pushbutton switch – a switch that can only be actuated in one way. A Green Power pushbutton switch is typically a momentary switch.

[11]**Subsequent commissioning** – ability to successfully complete commissioning exchange for an already commissioned GPDF, without prior reset.

[11] Dec 2016 SVE comment: https://workspace.zigbee.org/kws/groups/PRO_GP/comments/view_comment?comment_id=1012

398

4 Acronyms and abbreviations

ACK	Acknowledgement
AIB	Application support layer Information Base
APDU	Application Protocol Data Unit
APS	Application Support Sub-layer
BTT	Broadcast Transaction Table
cGP	Common Green Power stub
dGP	Dedicated Green Power stub
dLPED	Dedicated Low Power End Device stub
GP	Green Power
GPC	Green Power Combo device
GPCB	Green Power Combo Basic device
GPCm	Green Power Combo Minimum device
GPCT	Green Power Commissioning Tool device
GPD	Green Power Device
GPEP	Green Power End Point
GPDF	Green Power Device Frame
GPD ID	Green Power Device Identifier
GPFS	Green Power Frame Sequence
GPM	Green Power Manager
GPP	Green Power Proxy device
GPPB	Green Power Proxy Basic
GPS	Green Power Sink device
GPT	Green Power Target device
GPT+	Green Power Target Plus device
HMAC	Keyed Hash Message Authentication Code
LPED	Low Power End Device
LSB	Least Significant Byte
MAC	Medium Access Control layer
MIC	Message Integrity Code
MPDU	MAC Protocol Data Unit
NPDU	Network Protocol Data Unit
PAN	Personal Area Network
SAP	Service Access Point
SrcID	GPD Source identifier
ZCL	Zigbee Cluster Library
ZED	Zigbee End Device
ZR	Zigbee Router
ZBA	Zigbee Commercial Building Automation application profile
ZHA	Zigbee Home Automation application profile
ZSE	Zigbee Smart Energy application profile

5 Certification status

Section 3.1 and 3.2 of the Green Power Proxy Basic PICS document [6] provide an overview of GP certifiable and non-certifiable functionality.

6 Overview

The goal of this specification is to allow for usage of energy-harvesting devices within the Zigbee eco-system.

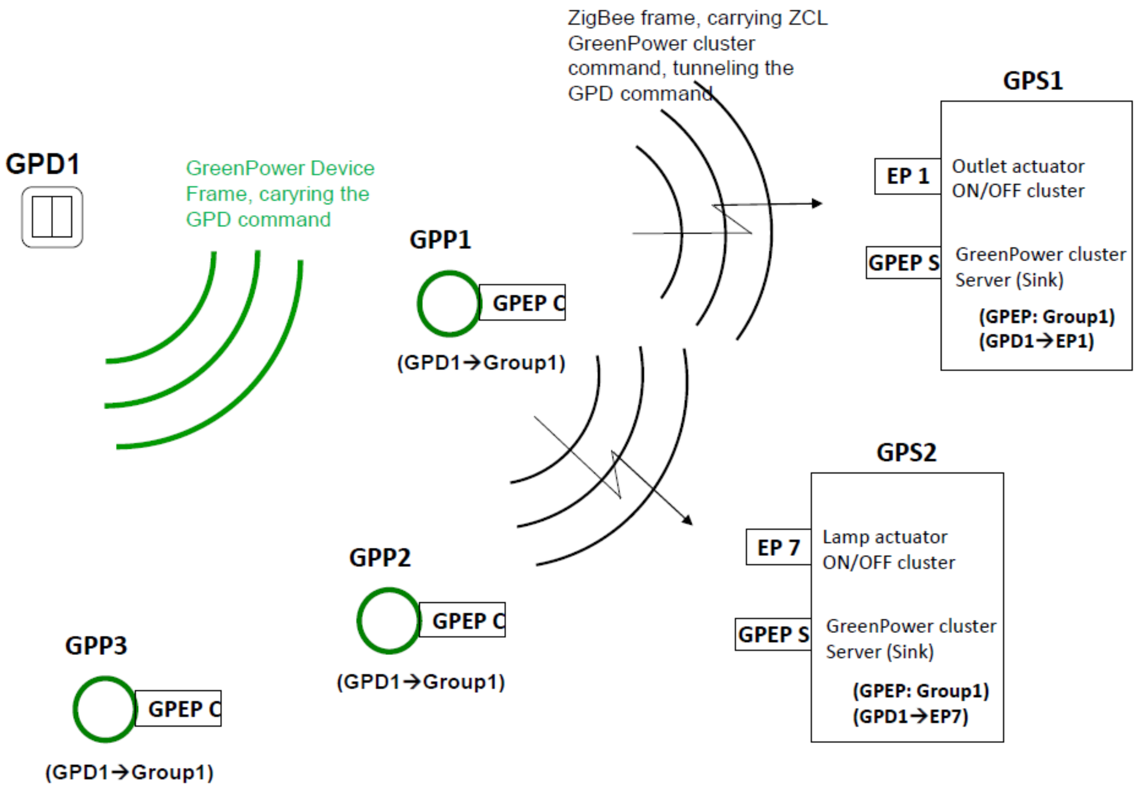
Such Green Power Devices, GPD, MAY harvest different [12]amounts of energy depending on the harvesting technology used. With its own available energy budget, each GPD has special requirements regarding the functionality it can implement. This specification defines different options which MAY be implemented by GPD depending on its energy budget, manufacturer choices and also profiles requirements.

Since GPD have very limited energy budget, the standard association-based two-way communication model of Zigbee is not readily applicable. To enable GPD to communicate to Zigbee network, this specification defines a new frame format for GPD (see sec. A.1.4), referred to as Green Power Device Frame (GPDF), much shorter than the Zigbee frame.

On the Zigbee network side, this specification defines the GP functionality required on a Zigbee node in order to receive and process the GPDF, and then tunnel it, if required – across multiple hops, in a normal Zigbee frame format to the paired to-be-controlled node, referred to as the sink, which processes and acts upon the information sent by GPD. That GP functionality is GP stub (section A.1) and Green Power cluster (section A.3), respectively.

This specification provides a way to commission GPD into a Zigbee network in order to pair GPD with the to-be-controlled nodes (section A.3.9).

Figure 1 provides a system overview for the networks involving Green Power devices.


ZigBee frame, carrying ZCL GreenPower cluster command, tunneling the GPD command
GPS1
GPD1
GreenPower Device Frame, carying the GPD command
EP 1
Outlet actuator ON/OFF cluster
GPP1
GPEP C
GPEP S
GreenPower cluster Server (Sink)
(GPEP: Group1)
(GPD1→EP1)
(GPD1→Group1)
GPS2
EP 7
Lamp actuator ON/OFF cluster
GPP2
GPEP C
GPEP S
GreenPower cluster Server (Sink)
(GPD1→Group1)
(GPEP: Group1)
(GPD1→EP7)
GPP3
GPEP C
(GPD1→Group1)

[12] LB v07: https://workspace.zigbee.org/kws/groups/zigbee_pro_foundation/comments/view_comment?comment_id=316

Figure 1 – System overview for the Green Power feature

The Green Power solution relies on the fact, that the future generation of Green Power sinks to be controlled by the GPD, implements the server side of the Green Power cluster, to interpret and act upon selected GPD commands. This architectural choice allows for simple operation of the Green Power proxy devices, which only have to tunnel the received GPDF to the sink, without translating it into a proper ZCL command. This makes the proxies application- and profile-agnostic and thus [13]forward-compatible with any future GPD types.

The sinks manage their own pairings, and propagate to the proxies only the relevant information, required for the tunneling. There is no fixed parent for the GPD; all proxies compete for the forwarding per packet. Thus, tunneling works in a fully distributed, self-organizing manner, while providing redundancy and reliability for the communication with GPD.

[13] LB v07: https://workspace.zigbee.org/kws/groups/zigbee_pro_foundation/comments/view_comment?comment_id=317

7 Candidate ZCL material for use with this specification

The candidate material in section A.3 MAY be merged into the Zigbee Cluster Library (ZCL) [3] by the Cluster Library Development Board.

The new cluster to be included in the ZCL has been allocated the ClusterID indicated in Table 1 by the Cluster Library Development Board (see also [11]).

Table 1 – Clusters ID allocation for candidate clusters

Functional Domain	Cluster Name	Provisional ClusterID	Where specified
General	Green Power cluster	0x0021	A.3

A.1 Green Power stub

A.1.1 Overview

Figure 2 shows a schematic view of how the GP communication mechanism works within a Zigbee stack. GP data exchanges are handled by a dedicated "stub", which is similar to the one specified in the ZSE profile for Inter-PAN.
The Common GP (cGP) stub performs the basic functions shared by LPED and GP. It performs just enough processing to pass application data frames to the MAC layer for transmission and to pass GPDF payload from the MAC to the relevant dedicated stub on receipt. The cGP stub is accessible to the higher layers through two special Service Access Point (SAP), CGP-SAP and CZLPED-SAP.
The dedicated LPED (dLPED) stub, as well as the corresponding LPED-SAPs, are out of scope of this document and will be defined separately by the Low Power End Device Task Group.
The dedicated GP (dGP) stub performs just enough processing to pass application data frames to the cGP stub for transmission and to pass GPD commands from the cGP stub to the Green Power cluster on Green Power EndPoint on receipt. The dGP stub is accessible to the higher layers through a special Service Access Point (SAP), GP-SAP, parallel to the normal APSDE-SAP. The dGP communication architecture does not support simultaneous execution by multiple application entities. A Zigbee router is assumed to have only one proxy application entity (Green Power EndPoint) that will use the GP communication mechanism.

The Green Power cluster SHALL be implemented on the reserved Green Power End Point - endpoint 0xF2 (242).

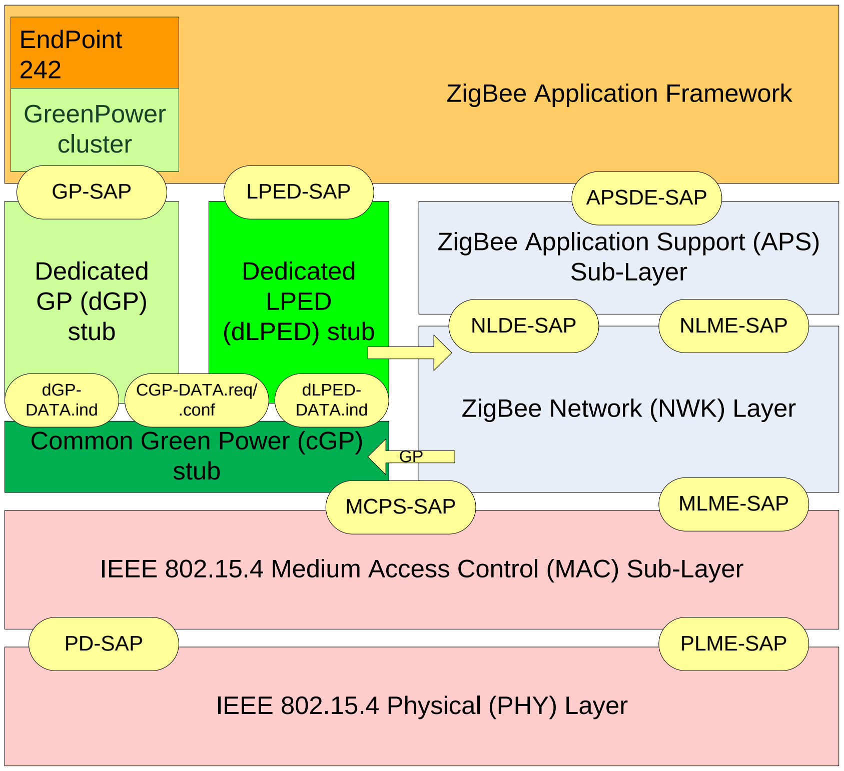

EndPoint 242
ZigBee Application Framework
GreenPower cluster
GP-SAP
LPED-SAP
APSDE-SAP
Dedicated GP (dGP) stub
Dedicated LPED (dLPED) stub
ZigBee Application Support (APS) Sub-Layer
NLDE-SAP
NLME-SAP
ZigBee Network (NWK) Layer
dGP-DATA.ind
CGP-DATA.req/ .conf
dLPED-DATA.ind
Common Green Power (cGP) stub
GP
MCPS-SAP
MLME-SAP
IEEE 802.15.4 Medium Access Control (MAC) Sub-Layer
PD-SAP
PLME-SAP
IEEE 802.15.4 Physical (PHY) Layer

Figure 2 – Zigbee Stack with the Green Power feature

The support of the GP feature, if provided, includes a couple of elements that require special attention. This is because they are so deep in or so tightly entangled with the Zigbee stack that for most implementations they would have to be provided by the stack vendor. Those include:

- The ability of a device implementing GP stub functionality (all GP infrastructure devices, except for GPT) to pass the frames with Zigbee protocol version 0x3 to the GP stub;
- The ability of a device implementing a GP proxy functionality to send a Zigbee frame with an alias NWK source address and alias NWK sequence number, and alias APS counter supplied by the Green Power EndPoint;
- The ability of Green Power EndPoint to act upon Device_annce and generate Device_annce for aliases;
- If bidirectional communication is to be supported by the GP infrastructure device, the ability to:
  - send GPDF at the time defined by the GP specification, including skipping CSMA/CA;
  - pass the MCPS-DATA.confirm returned by the MAC layer to the appropriate protocol stack;
- If LPED functionality is to be supported: the NWKLPED-DATA.indication primitive.

It is recommended though that the stack vendors to implement the complete GP feature – and certify it as part of the Zigbee Compliant Platform certification.

However, the GP code can be built by anybody, if the elements listed above are provided. Therefore, the stack vendors that do not intend to provide the full GP implementation are recommended to consider providing those elements as compliable components.

A.1.2 cGP stub

The cGP stub is responsible for the GPDF packet formation and parsing, as well as the following filtering tasks: simple duplicate filtering, dropping of the GPDF based of the *Direction* sub-field of the *Extended NWK Frame Control* field, and filtering and de-multiplexing based on the *ApplicationID* sub-field of the *Extended NWK Frame Control* field.

A.1.2.1 cGP stub Service Specification

The CGP-SAP is a data service comprising the following primitives shared by the dGP and dLPED stubs:

- CGP-DATA.request – provides a mechanism for dGP stub or dLPED stub to request cGP stub to transmit a GPDF.
- CGP-DATA.confirm – provides a mechanism for dGP stub or dLPED stub to understand the status of a previous request to send a GPDF.

The dGP-SAP is a data service comprising the following primitives:

- dGP-DATA.indication – provides a mechanism for cGP stub to identify and convey a received GPDF to dGP stub.

The dLPED-SAP is a data service comprising the following primitives:

- CLPED-DATA.indication – provides a mechanism for cGP stub to identify and convey a received LPED GPDF to dLPED stub.

A.1.2.1.1 CGP-DATA.request

A.1.2.1.1.1 Semantics of the CGP-DATA.request primitive

```
CGP-DATA.request          {
                          TxOptions
                          SrcAddrMode,
                          SrcPANId,
                          SrcAddr,
DstAddrMode,
DstPANId,
DstAddr,
GP MPDU Length
GP MPDU
GP MPDU Handle
}
```

Table 2 – Parameters of the CGP-DATA.request

Name	Type	Valid Range	Description
TxOptions	8-bit bitmap	Any Valid	The transmission options for this GPDF. These are a bitwise OR of one or more of the following: 0x01 = Use CSMA/CA 0x02 = Use MAC ACK 0x04 – 0xff - reserved
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.
SrcPANId	16-bit PAN Id	0x0000 – 0xffff	The 16-bit PAN identifier of the entity sending this MPDU.
SrcAddress	16-bit or 64-bit address	As specified by the SrcAddrMode parameter	The device address of the entity sending this MPDU.
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: 0 x 00 = no address (DstPANId and DstAddr omitted) 0x01 = reserved 0x02 = 16-bit NWK address, normally 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 MPDU is being transferred or the broadcast PAN ID 0xffff.
DstAddr	16-bit or 64-bit address	As specified by the DstAddrMode parameter	The address of the entity to which the MPDU is being transferred or the broadcast address 0xffff.
GP MPDU Length	Integer	0x00 – (*aMaxMACFrameSize* - 9)	The number of octets in the transmitted GP MPDU.
GP MPDU	Sequence of octets	-	The sequence of octets forming the transmitted GP MPDU. It SHALL be the full MPDU, as defined in A.1.4.1.
GP MPDU Handle	Unsigned 8-bit integer	0x00-0xff	The handle used between the dGP/dLPED stub and the cGP stub, to match the request with the confirmation.

A.1.2.1.1.2 When generated

This primitive is generated by the dGP or the dLPED stub when a GPDF is to be sent to the GPD /LPED identified by the *DstAddr*.

A.1.2.1.1.3 Effect on receipt

Upon receipt of this primitive the CGP stub SHALL send the MPDU to the MAC layer for transmission.

The parameter *UseCSMA* of the *TxOptions* is an extension to the MCPS-DATA.request and SHALL be propagated by the cGP stub to the MAC layer. When *UseCSMA* is FALSE, CSMA/CA SHALL be skipped for the transmission of this GPDF.

A.1.2.1.2 CGP-DATA.confirm

A.1.2.1.2.1 Semantics of the CGP-DATA.confirm primitive

```
CGP-DATA.confirm {
                                        Status
                                        GP MPDU handle
                                        }
```

Table 3 – Parameters of the CGP-DATA.confirm

Name	Type	Valid Range	Description
Status	Enumeration	Any valid	Status code, as returned by the MAC layer (see Table 28 of [16]).
GP MPDU handle	Unsigned 8-bit integer	0x00-0xff	The handle used between dGP/dLPED stub and cGP stub, to match the request with the confirmation.

A.1.2.1.2.2 When generated

This primitive is generated by the cGP stub and passed to the dGP stub/dLPED stub after the CGP-DATA.request has been handled.

A.1.2.1.2.3 Effect on receipt

Upon receipt of this primitive the dGP/dLPED stub is informed about the status of its request to transmit a GPDF, as indicated by the GP MPDU handle.

A.1.2.1.3 dGP-DATA.indication primitive

A.1.2.1.3.1 Semantics of the dGP-DATA.indication primitive

```
dGP-DATA.indication             {
                                        RSSI
                                        Link Quality
                                        SeqNumber
                                        SrcAddrMode
                                        SrcPANId
                                        SrcAddress
                                        DstAddrMode
                                        DstPANId
                                        DstAddress
                                        GP MPDU Length
                                        GP MPDU
                                        }
```

Table 4 – Parameters of the dGP-DATA.indication

Name	Type	Valid Range	Description
RSSI	signed 8-bit integer	0x00 – 0xff	The RSSI delivered by the MAC on receipt of this frame.
Link quality	unsigned 8-bit integer	0x00 – 0xff	The LQI delivered by the MAC on receipt of this frame.
SeqNumber	Unsigned 8-bit integer	0x00 – 0xff	The sequence number from MAC header of the received MPDU.
SrcAddrMode	Integer	0x00 – 0x03	The source addressing mode for this primitive corresponding to the received MPDU. 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.
SrcPANId	16-bit PAN Id	0x0000 – 0xffff	The 16-bit PAN identifier of the GPD entity from which the ASDU was received.
SrcAddress	16-bit or 64-bit address	As specified by the SrcAddrMode parameter	The device address of the GPD entity from which the ASDU was received.
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: 0 x 00 = no address (DstPANId and DstAddress omitted) 0x01 = reserved 0x02 = 16-bit NWK address, normally 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 or the broadcast address 0xffff.
GP MPDU Length	Integer	0x00 – (*aMaxMACFrameSize* - 9)	The number of octets in the received GP MPDU.
GP MPDU	Sequence of octets	-	The sequence of octets forming the received GP MPDU.

A.1.2.1.3.2 When generated

This primitive is generated and passed to the dGP stub in the event of the receipt, by the cGP stub, of a MCPS-DATA.indication primitive from the MAC sub-layer, containing a GPDF with *ApplicationID* sub-field 0b000 or 0b010 and *Direction* sub-field 0b0.

A.1.2.1.3.3 Effect on receipt

Upon receipt of this primitive the dGP stub is informed of the receipt of a GPDF transmitted, via the cGP stub, by a GPD device and intended for the receiving device.

A.1.2.1.4 dLPED-DATA.indication primitive

A.1.2.1.4.1 Semantics of the dLPED-DATA.indication primitive

The dLPED-DATA.indication primitive is formatted exactly as the dGP-DATA.indication primitive (see sec. A.1.2.1.3.1).

A.1.2.1.4.2 When generated

This primitive is generated and passed to the dLPED stub in the event of the receipt, by the cGP stub, of a MCPS-DATA.indication primitive from the MAC sub-layer, containing a GPDF with *ApplicationID* sub-field 0b001 (LPED).

A.1.2.1.4.3 Effect on receipt

Upon receipt of this primitive the dLPED stub is informed of the receipt of an LPED GPDF transmitted, via the cGP stub, by a peer device and intended for the receiving device.

A.1.3 dGP stub Service Specification

The GP-SAP is a data service comprising the following primitives:

- GP-DATA.request – provides a mechanism for the Green Power EndPoint to request transmission of a GPDF.
- GP-DATA.confirm – provides a mechanism for the Green Power EndPoint to understand the status of a previous request to send a GPDF.
- GP-DATA.indication – provides a mechanism for identifying and conveying a received GPDF to the Green Power EndPoint.
- GP-SEC.request – provides a mechanism for dGP stub to request security data from the Green Power EndPoint.
- GP-SEC.response – provides a mechanism for the Green Power EndPoint to provide security data into the dGP stub.

A.1.3.1 GP-DATA.indication primitive

A.1.3.1.1 Semantics of the GP-DATA.indication primitive

```
GP-DATA.indication        {
                          Status
                          RSSI
                          Link Quality
                          SeqNumber
                          SrcAddrMode
                          SrcPANId
                          SrcAddress
                          ApplicationID
                          GPDFSecurityLevel
                          GPDFKeyType
                          AutoCommissioning
                          RxAfterTx
                          SrcID
                          Endpoint
                          GPD security frame counter
                          GP CommandID
                          GP ASDU Length
                          GP ASDU
                          MIC
                          }
```

610 **Table 5 – Parameters of the GP-DATA.indication**

Name	Type	Valid Range	Description
Status	8-bit enumeration	Any valid	Status code, as returned by dGP stub. It can have the following values: SECURITY_SUCCESS NO_SECURITY COUNTER_FAILURE AUTH_FAILURE UNPROCESSED
RSSI	signed 8-bit integer	0x00 – 0xff	The RSSI delivered by the MAC on receipt of this frame.
Link quality	unsigned 8-bit integer	0x00 – 0xff	The LQI delivered by the MAC on receipt of this frame.
SeqNumber	Unsigned 8-bit integer	0x00 – 0xff	The sequence number from MAC header of the received MPDU.
SrcAddrMode	8-bit enumeration	0x00 – 0x03	The source addressing mode for this primitive corresponding to the received MPDU. 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.
SrcPANId	16-bit PAN Id	0x0000 – 0xffff	The 16-bit PAN identifier of the GPD entity from which the ASDU was received.
SrcAddress	16-bit or 64-bit address	As specified by the SrcAddrMode parameter	The device address of the GPD entity from which the ASDU was received.
ApplicationID	8-bit enumeration	0x00, 0x02	The *ApplicationID*, corresponding to the received MPDU. *ApplicationID* 0x00 indicates the usage of the SrcID; *ApplicationID* 0x02 indicates the usage of the GPD IEEE address.
GPDFSecurityLevel	8-bit enumeration	0x00, 0x02 – 0x03	The security level, corresponding to the received MPDU.
GPDFKeyType	8-bit enumeration	0x00 - 0x07	The security key type, which was successfully used for security processing the received MPDU.
Auto-Commissioning	Boolean	TRUE/FALSE	The Auto-Commissioning sub-field, copied from the received GPDF.
RxAfterTx	Boolean	TRUE/FALSE	The *RxAfterTx* sub-field, copied from the received GPDF.
[14]SrcID	Unsigned 32-bit Integer	0x00000000 – 0xffffffff	The identifier of the GPD entity from which the ASDU was received. If the *Frame Type* sub-field of the received GPDF was set to 0b01, the SrcID parameter SHALL carry the value 0x00000000. If the *Frame Type* sub-field of the received GPDF was set to 0b00 and the *ApplicationID* sub-field of the received GPDF was set to 0b000 or absent, the SrcID parameter SHALL carry the value copied from the *GPD SrcID* field of the triggering GPDF. If the *ApplicationID* sub-field of the received GPDF was set to 0b010, the SrcID parameter is ignored.
Endpoint	Unsigned 8-bit integer	0x00 – 0xf0, 0xff	The identifier of the GPD endpoint used in combination with the GPD IEEE address if *ApplicationID* = 0b010. If *ApplicationID* = 0b000 this parameter is ignored.
GPD security frame counter	Unsigned 32-bit Integer	As specified by the GPDFSecurityLevel parameter	The security frame counter value used on transmission by the GPD entity from which the ASDU was received.

[14] CCB #2360; Resolution added in 15-02014-011

GPD Command ID	Unsigned 8-bit integer	0x00 – 0xff	The identifier of the command, within the GP specification, which defines the application semantics of the ASDU.
GPD ASDU Length	Unsigned 8-bit integer	0x00 – (*aMaxMACFrameSize* - 9)	The number of octets in the received GPD ASDU.
GPD ASDU	Sequence of octets	-	The sequence of octets forming the received GPD ASDU.
MIC	Unsigned 16-bit or 32-bit Integer	As specified by the GPDFSecurityLevel parameter	The sequence of octets forming the MIC for the received GPD MPDU.

A.1.3.1.2 When generated

[15]This primitive is generated and passed to the application in the event of the receipt, by the dGP stub, of a dGP-DATA.indication primitive from cGP, containing a frame that was generated by the GPD, and that was intended for the receiving device.
The reasons for the various *Status* codes are described in sec. A.1.5.2.2.

A.1.3.1.3 Effect on receipt

Upon receipt of this primitive the application is informed of the receipt of an application frame transmitted, via the dGP stub, by a peer device and intended for the receiving device.

A.1.3.2 GP-DATA.request

A.1.3.2.1 Semantics of the GP-DATA.request primitive

```
GP-DATA.request    {
                   Action
                   TxOptions
                   ApplicationID
                   SrcID
                   GPD IEEE address
                   Endpoint
                   GPD CommandID
                   GPF ASDU Length
                   GPD ASDU
                   GPEP handle
                   gpTxQueue Entry Lifetime
                   }
```

[15] CCB #2424; Resolution added in GP Basic spec errata 15-02014-011

Table 6 – Parameters of the GP-DATA.request

Name	Type	Valid Range	Description
Action	Boolean	TRUE/FALSE	TRUE: add GPDF into the queue FALSE: remove GPDF from queue
TxOptions	8-bit bitmap	Any Valid	The transmission options for this GPDF. These are a bitwise OR of one or more of the following: b0 = Use gpTxQueue b1 = Use CSMA/CA b2 = Use MAC ACK b3-b4 = GPDF frame type for Tx (can take non-reserved values as defined in Table 10) b5 = Tx on matching endpoint b6 – b7 – reserved
ApplicationID	8-bit enumeration	0x00, 0x02	*ApplicationID* of the GPD to which the ASDU will be sent; *ApplicationID* 0x00 indicates the usage of the SrcID; *ApplicationID* 0x02 indicates the usage of the GPD IEEE address.
[16]SrcID	Unsigned 32-bit Integer	0x00000000 – 0xffffffff	The identifier of the GPD entity to which the ASDU will be sent if *ApplicationID* = 0b000. If the Frame Type sub-field of the TxOptions parameter is set to 0b01, the SrcID parameter SHALL carry the value 0x00000000. If the Frame Type sub-field of the TxOptions parameter is set to 0b00 and the *ApplicationID* parameter is set to 0b000, the SrcID parameter SHALL carry the value to be copied into the *GPD SrcID* field of the to be transmitted GPDF. If the *ApplicationID* parameter is set to 0b010, the SrcID parameter is ignored.
GPD IEEE address	IEEE address	Any valid	The identifier of the GPD entity to which the ASDU will be sent if *ApplicationID* = 0b010.
Endpoint	Unsigned 8-bit integer	0x00 – 0xf0, 0xff	The identifier of the GPD endpoint used in combination with the GPD IEEE address if ApplicationID = 0b010. If *ApplicationID* = 0b000 this parameter is ignored.
GPD Command ID	Integer	0x00 – 0xff	The identifier of the command, within the GP specification, which defines the application semantics of the ASDU.
GPD ASDU Length	Integer	0x00 – (*aMaxMACFrameSize* - 9)	The number of octets in the transmitted GPD ASDU.
GPD ASDU	Sequence of octets	-	The sequence of octets forming the transmitted GPD ASDU.
GPEP handle	Unsigned 8-bit integer	0x00-0xff	The handle used between Green Power EndPoint and dGP stub, to match the request with the confirmation.
gpTxQueueEntry-Lifetime	Unsigned 24-bit integer	0x000000 – 0xffffff	The lifetime of this packet in the gpTxQueue, in milliseconds. 0x000000 indicates immediate transmission. 0xffffff indicates infinity. In a Basic Proxy/Sink, the default lifetime MAY be 0xffffff.

A.1.3.2.2 When generated

This primitive is generated by the Green Power EndPoint and passed to the dGP stub when a GPDF is to be sent to the GPD identified by the GPD SrcID or GPD IEEE address and Endpoint.

[16] CCB #2360; Resolution added in 15-02014-011

A.1.3.2.3 Effect on receipt

Upon receipt of this primitive with the *Action* parameter is set to TRUE, the dGP stub SHALL add the GPDF to the gpTxQueue and store all the relevant data, including the *GPD ID*, *Endpoint* if *ApplicationID* = 0b010 and *TxOptions*. If *ApplicationID* = 0b010 and the *Tx on matching endpoint* sub-field of the *TxOptions* parameter has the value of 0b0, then any existing gpTxQueue entry for this GPD IEEE address SHALL be removed, irrespective of the value of the *Endpoint* field of the queue entry and *Endpoint* parameter of the primitive. If *ApplicationID* = 0b010 and the *Tx on matching endpoint* sub-field of the *TxOptions* parameter has the value of 0b1, then only existing gpTxQueue entries storing *Endpoint* field 0xff or equal to the *Endpoint* parameter from the primitive SHALL be removed.

Upon receipt of this primitive with the *Action* parameter is set to FALSE, the dGP stub SHALL remove the gpTxQueue entry as indicated by the *GPD ID* and, if *ApplicationID* = 0b010, *Endpoint* parameters.

A.1.3.3 GP-DATA.confirm

A.1.3.3.1 Semantics of the GP-DATA.confirm primitive

```
GP-DATA.confirm    {
                   Status
                   GPEP handle
                   }
```

Table 7 – Parameters of the GP-DATA.confirm

Name	Type	Valid Range	Description
Status	Enumeration	Any valid	Status code, as returned by the CGP stub. In addition to the values returned by the MAC layer, it can have the following values: TX_QUEUE_FULL ENTRY_REPLACED ENTRY_ADDED ENTRY_EXPIRED ENTRY_REMOVED GPDF_SENDING_FINALIZED
GPEP handle	Unsigned 8-bit integer	0x00-0xff	The handle used between Green Power EndPoint and the lower layers, to match the request with the confirmation.

A.1.3.3.2 When generated

This primitive is generated by the lower layers and passed to the Green Power EndPoint after the GP-DATA.request has been handled.
The reasons for the various *Status* codes are described in sec. A.1.5.2.1.

A.1.3.3.3 Effect on receipt

Upon receipt of this primitive the Green Power EndPoint is informed about the status of its request to transmit data to GPD, as indicated by the GPEP handle.

A.1.3.4 GP-SEC.request

A.1.3.4.1 Semantics of the GP-SEC.request primitive

```
GP-SEC.request     {
                   ApplicationID
                   SrcID
                   GPD IEEE address
```

```
                         Endpoint
                         GPDFSecurityLevel
                         GPDFKeyType
                         GPDSecurityFrameCounter
                         dGP stub handle
                         }
```

Table 8 – Parameters of the GP-SEC.request

Name	Type	Valid Range	Description
ApplicationID	8-bit enumeration	0x00, 0x02	*ApplicationID* of the GPD entity from which the ASDU was received. *ApplicationID* 0x00 indicates the usage of the SrcID; *ApplicationID* 0x02 indicates the usage of the GPD IEEE address.
[17]SrcID	Unsigned 32-bit Integer	0x00000000 – 0xffffffff	The identifier of the GPD entity from which the ASDU was received if *ApplicationID* = 0b000.
GPD IEEE address	IEEE address	Any valid	The identifier of the GPD entity from which the ASDU was received if *ApplicationID* = 0b010.
Endpoint	Unsigned 8-bit integer	0x00 – 0xf0, 0xff	The identifier of the GPD endpoint used in combination with the GPD IEEE address if *ApplicationID* = 0b010. If *ApplicationID* = 0b000 this parameter is ignored.
GPDFSecurityLevel	8-bit enumeration	0x00, 0x02 – 0x03	The security level, corresponding to the received MPDU.
GPDFKeyType	8-bit enumeration	0x00 - 0x01	The security key type, corresponding to the received MPDU.
GPD security frame counter	Unsigned 8-bit or 32-bit Integer	As specified by the *GPDFSecurityLevel* parameter	The security frame counter value corresponding to the received MPDU.
dGP stub handle	Unsigned 8-bit integer	0x00-0xff	The handle used between dGP stub and the higher layers, to match the request with the response.

A.1.3.4.2 When generated

This primitive is generated by the dGP stub and passed to the Green Power EndPoint on reception of protected GPDF.

A.1.3.4.3 Effect on receipt

Upon receipt of this primitive the Green Power EndPoint is informed about reception of protected GPDF. The Green Power EndPoint responds with GP-SEC.response primitive, with appropriate status, based on the Green Power EndPoint client/server functionality, the operational/commissioning mode the Green Power EndPoint is in and the content of Proxy/Sink Table.

A.1.3.5 GP-SEC.response

A.1.3.5.1 Semantics of the GP-SEC.response primitive

```
GP-SEC.response          {
                         Status
                         dGP stub handle
                         ApplicationID
                         SrcID
                         GPD IEEE address
                         Endpoint
                         GPDFSecurityLevel
```

[17] CCB #2360; Resolution added in 15-02014-011

```
                    GPDFKeyType
                    GPDKey
                    GPDSecurityFrameCounter
                    }
```

Table 9 – Parameters of the GP-SEC.response

Name	Type	Valid Range	Description
Status	8-bit enumeration	Any valid	The status code, as returned by the Green Power EndPoint. The following are supported: MATCH DROP_FRAME PASS_UNPROCESSED TX_THEN_DROP
dGP stub handle	Unsigned 8-bit integer	0x00-0xff	The handle used between dGP stub and the higher layers, to match the request with the response.
ApplicationID	8-bit enumeration	0x00, 0x02	*ApplicationID* of the GPD entity from which the ASDU was received. *ApplicationID* 0x00 indicates the usage of the SrcID; *ApplicationID* 0x02 indicates the usage of the GPD IEEE address.
[18]SrcID	Unsigned 32-bit Integer	0x00000000 – 0xffffffff	The identifier of the GPD entity from which the ASDU was received if *ApplicationID* = 0b000.
GPD IEEE address	IEEE address	Any valid	The identifier of the GPD entity from which the ASDU was received if *ApplicationID* = 0b010.
Endpoint	Unsigned 8-bit integer	0x00 – 0xf0, 0xff	The identifier of the GPD endpoint used in combination with the GPD IEEE address if *ApplicationID* = 0b010. If *ApplicationID* = 0b000 this parameter is ignored.
GPDFSecurityLevel	8-bit enumeration	0x00, 0x02 – 0x03	The security level to be used for GPDF security processing.
GPDFKeyType	8-bit enumeration	0x000 - 0x07	The security key type to be used for GPDF security processing.
GPD Key	Security Key	Any valid	The security key to be used for GPDF security processing.
GPD security frame counter	Unsigned 32-bit Integer	Any valid	The security frame counter value to be used for GPDF security processing.

A.1.3.5.2 When generated

This primitive is generated by the Green Power EndPoint and passed to the dGP stub on reception of GP-SEC.request.

A.1.3.5.3 Effect on receipt

Upon receipt of this primitive the dGP stub checks the value of the *Status* field. If the *Status* is MATCH or TX_THEN_DROP, the dGP stub triggers security processing of the GPDF, with the supplied parameters. If the *Status* is DROP_FRAME, it silently drops the frame. If the *Status* is PASS_UNPROCESSED, it generates GP-DATA.indication with the [19]*Status* UNPROCESSED, and with unprocessed fields GPD CommandID, GPD Command Payload and MIC copied from the received GPDF.

A.1.3.6 NWKLPED-DATA.indication

This primitive requests the transfer of a data PDU (NSDU) from the dLPED stub to a single or multiple peer APS sub-layer entities.

[18] CCB #2360; Resolution added in 15-02014-011
[19] CCB #2362; Resolution added in 15-02014-011

The parameters of the NWKLPED-DATA parameters consist of an NWK header and NWK payload as described in section 3.3.1 “General NPDU Frame Format” of [1].

A.1.3.6.1 When generated

This primitive is generated by the local dLPED stub whenever a data PDU (NSDU) is to be transferred to a single or multiple peer APS sub-layer entity.

A.1.3.6.2 Effect on receipt

If this primitive is received the NWK layer SHALL process it as if it were an incoming frame received via NLDE-DATA.indication already after incoming frame security processing, i.e. route the packet as defined in section 3.6.3 “Routing” of [1].

A.1.3.7 Green Power cluster

Please note, that the Green Power cluster, when sending ZCL commands via Zigbee stack, provides the parameters *UseAlias*, *SrcAddr* and *NWKSeqNumb*, as an extension to the APSDE-DATA.request and NLDE-DATA.request. They SHALL be propagated by the Zigbee APS sub-layer to the NWK layer.

The supplied *UseAlias*, if set to 0b1, indicates that the supplied *SrcAddr* and *NWKSeqNumb* parameters SHALL be used; otherwise they can be ignored.

When *UseAlias* is set to 0b1, the supplied *SrcAddr* SHALL be used in the NWK header *SrcAddress* field, instead of the device’s own short address, as stored in the NIB *nwkNetworkAddress parameter*. The NIB *nwkNetworkAddress* SHALL NOT be changed.

When *UseAlias* is set to 0b1, the supplied *NWKSeqNumb* SHALL be used in the NWK header *SeqNumber* field, instead of the NWK-maintained *nwkSequenceNumber* parameter of the NIB and in the APS header *APS counter* field, instead of the APS-maintained counter value. The NIB *nwkSequenceNumber* and the APS-maintained counter SHALL NOT be overwritten.

A.1.4 Frame formats

The birds-eye view of a normal Zigbee frame as defined in [1] is shown in Figure 3. Briefly, the frame contains the 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 [3].

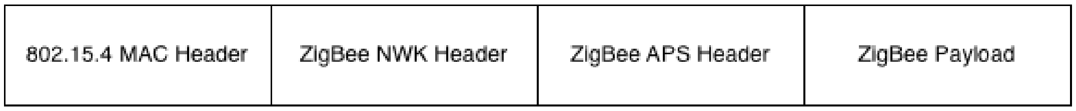


Figure 3 – Normal Zigbee Frame

Since most of the information contained in the NWK and all the information in the APS, headers is not relevant for GP operation, the GP frame contains a modified NWK header, and no APS header, followed by a dedicated application payload.

As for IEEE802.15.4 and Zigbee frames, all the Green Power frame fields SHALL be transmitted in little Endian.

A.1.4.1 Generic GPDF frame format

The GPDF frame has a generic format as illustrated in Figure 4 and Figure 5.

Octets: 2	1	4/10/12/variable	1	0/1	0/4	0/1	0/4
Frame Control	Sequence Number	Addressing fields	NWK Frame Control	Extended NWK Frame Control	GPD SrcID	Endpoint	Security frame counter
802.15.4 MAC Header			GP stub NWK Header				

Figure 4 – GPDF Frame Format (part 1)

Variable	0/4	2
GP Application Payload	MIC	FCS
GP Application Payload	GP stub NWK Trailer	802.15.4 MAC Trailer

Figure 5 – GPDF Frame Format (part 2)

A.1.4.1.1 MAC header fields

The MAC header fields SHALL be set such that the frame can be correctly received. Additional MAC fields, which are not strictly required for GPDF addressing, MAY be included both for *ApplicationID* 0b000 and 0b010, and both in the *Direction* to and from the GPD, as long as the frame remains 802.15.4-2003 [16] compliant; those additional fields SHALL be ignored upon reception and SHALL NOT be used for any further GPDF processing. Device vendors need to consider the inclusion of the additional fields carefully, since it increases packet airtime and energy consumption on the sender and receiver.

In order to allow for GPD mobility and make use of the built-in receiver redundancy, the GPDF originating from the GPD can be sent with MAC *Dest PANID* and MAC *Dest Address* set to 0xffff.

If the IEEE address of the GPD is used for unique identification of GPD, the GPDF SHALL include the *Extended NWK Frame Control* field and its *ApplicationID* sub-field SHALL be set to 0b010. Then, for the GPDF transmitted by the GPD, the GPD's IEEE address SHALL be transmitted in the MAC *Src Address* field, and the *Intra-PAN* sub-field and the *Source Addressing Mode* sub-field of the MAC *Frame Control* field SHALL be set accordingly. For the GPDF transmitted to the GPD, the GPD's IEEE address SHALL be transmitted in the MAC *Dest Address* field, and the *Intra-PAN* sub-field and the *Destination Addressing Mode* sub-field of the MAC *Frame Control* field SHALL be set accordingly.

In a Maintenance *Frame Type*, the IEEE address of the GPD SHOULD be omitted.

A.1.4.1.2 NWK Frame Control field

The *NWK Frame Control* field is formatted as shown in Figure 6.

Bits: 0-1	2-5	6	7
Frame type	Zigbee Protocol Version	Auto Commissioning	NWK Frame Control Extension

Figure 6 – Format of the NWK Frame Control field of GPDF

The *Zigbee Protocol Version* sub-field SHALL carry the value of 0x3.

The *Frame type* sub-field, as used in combination with the *Zigbee Protocol Version* = 0x3, can take the values as specified in Table 10.

Table 10 – Values of *Frame Type* used in combination with *Zigbee Protocol Version* = 0x3

Value	Description
0b00	Data frame
0b01	Maintenance frame
0b10	Reserved
0b11	Reserved

[20] Received GPDF with *Frame Type* other than 0b00 and 0b01 SHALL be dropped without further processing.

The *Auto-Commissioning* sub-field has different meaning in a Data (0b00) and Maintenance (0b01) *Frame Type*.

In a Data *Frame Type*, the *Auto-Commissioning* sub-field indicates if the GPD implements the Commissioning GPDF. If set to 0b1, the GPD does not implement the Commissioning GPDF. If set to 0b0, the GPD does implement the Commissioning GPDF.
A GPDF SHALL NOT have *RxAfterTx* sub-field of the *Extended NWK Frame Control* field and *Auto-Commissioning* field of *NWK Frame Control* field both set to 0b1; such a frame SHALL be silently dropped.

In a Maintenance *Frame Type*, the *Auto-Commissioning* sub-field, if set to 0b0, indicates that the GPD will enter the receive mode *gpdRxOffset* ms after completion of this GPDF transmission, for at least *gpdMinRxWindow*. If the value of this sub-field is 0b1, then the GPD will not enter the receive mode after sending this particular GPDF.

The *NWK Frame Control Extension*, if set to 0b1, indicates that the *Extended NWK Frame Control* field of the GPDF is present.

A.1.4.1.3 Extended NWK Frame Control field

The *Extended NWK Frame Control* field has the format as defined in Figure 7. It SHALL be present if the *ApplicationID* is different than 0b000.

Bits: 0-2	3-7
Application ID	Defined for specific ApplicationID

Figure 7 – Generic format of the Extended NWK Frame Control field of GPDF

The *ApplicationID* allows for re-defining the GPDF frame format. The current specification defines the GPDF frame format for *ApplicationID* 0b000 and 0b010 (GP) and *ApplicationID* 0b001 (LPED). Default value to be used on reception, if the *Extended NWK Frame Control* field is not present, is 0b000.

[21]According to the current specification, received GPDF with *ApplicationID* other than 0b000 and 0b010 SHALL be dropped without further processing.

The bits 3-7 of the *Extended NWK Frame Control* field are defined by *ApplicationID*.

For *ApplicationID* 0b000 and 0b010 (GP) and *ApplicationID* 0b001 (LPED), the bits 3-7 are defined in Figure 8. For *ApplicationID* 0b000[22], the *Extended NWK Frame Control* field SHALL be present if the GPDF is protected, if *RxAfterTx* is set, or if the GPDF is sent to the GPD.

[20] CCB #2325; Resolution added in 15-02014-011; Superballot comment #2189 December 2018, resolution added in 16-02607-027
[21] CCB #2325; Resolution added in 15-02014-011

Bits: 3-4	5	6	7
Security Level	Security Key	RxAfterTx	Direction

Figure 8 – Format of the Extended NWK Frame Control field for *ApplicationID* 0b000 and 0b010 (GP) and 0b001 (LPED)

The *SecurityLevel* sub-field indicates if the frame is protected [23]and which level of security is used to protect the current frame.

If *ApplicationID* is set to 0b000 or 0b010, the *SecurityLevel* sub-field can have values as defined in Table 11. Default value to be used on reception, if the *Extended NWK Frame Control* field is not present, is 0b00. If the *SecurityLevel* is set to 0b00, the *SecurityKey* sub-field is ignored on reception, and the fields *Security frame counter* and *MIC* are not present. The *MAC sequence number* field carries the random or the incremental sequence number, according to the capabilities of this GPD. If the *SecurityLevel* is set to 0b10 or 0b11, the *Security Frame counter* field is present, has the length of 4B, and carries the full 4B security frame counter, the *MIC* field is present, has the length of 4B, and carries the full 4B Message Integrity Code (see sec. A.1.5.3.4). The MAC *sequence number* field carries the random or the incremental sequence number, according to the capabilities of this GPD; it SHALL NOT be used for security, but only for duplicate filtering at MAC level.

If *ApplicationID* is set to 0b001, the *Security Level* sub-field SHALL be set to 0b10 or 0b11, the *Security Frame counter* field is present, and the *MIC* field is present, has the length of 4B, and carries the full 4B Message Integrity Code (see sec. A.1.5.3.4).

Table 11 – Values of gpSecurityLevel

Value	Description
0b00	No security
0b01	Reserved
0b10	4B frame counter and 4B MIC only
0b11	Encryption & 4B frame counter and 4B MIC

According to the current version of the specification, only GPD that support *gpdSecurityLevel* = 0b10 or higher AND support TC-LK protection (as indicated by the *GPDkeyEncryption* sub-field of the *Extended Options* field of the GPD Commissioning command) of the GPD key, if exchanged over the air, can be certified.

The *SecurityKey* sub-field indicates the type of the key used for [24]protection of this frame. The mapping between the gpSecurityKeyType used for the GPDF protection and the value of the *SecurityKey* sub-field as indicated in the *Extended NWK Frame Control* field of the GPDF is defined in Table 12.

Table 12 – Mapping between the *gpSecurityKeyType* and the *SecurityKey* sub-field of the *Extended NWK Frame Control* field

gpSecurityKeyType	Corresponding value of the SecurityKey sub-field of the GPDF Extended NWK Frame Control field
0b000	0b0

[22] CCB #2422; resolution added in 15-02014-011
[23] CCB #2421; resolution added in 15-02014-011
[24] CCB #2415, resolved in 15-02014r010

gpSecurityKeyType	Corresponding value of the SecurityKey sub-field of the GPDF Extended NWK Frame Control field
0b001	0b0
0b010	0b0
0b011	0b0
0b100	0b1
0b101-0b110	Reserved
0b111	0b1

The *RxAfterTx* sub-field is a Boolean flag. If the value of this sub-field is 0b1, then it indicates that the GPD will enter the receive mode *gpdRxOffset* ms after completion of this GPFS transmission, for at least *gpdMinRxWindow*. If the value of this sub-field is 0b0, then the GPD will not enter the receive mode after sending this particular GPFS. Default value to be used on reception, if the *Extended NWK Frame Control* field is not present, is 0b0.
A GPDF SHALL NOT have *RxAfterTx* sub-field of the *Extended NWK Frame Control* field and *Auto-Commissioning* field of *NWK Frame Control* field both set to 0b1; such a frame SHALL be silently dropped.

The *Direction* sub-field SHALL be set to 0b0, if the GPDF is transmitted by the GPD, and to 0b1, if the GPDF is transmitted by a proxy. Default value to be used on reception, if the *Extended NWK Frame Control* field is not present, is 0b0.

A.1.4.1.4 GPD SrcID field

The *GPDSrcID* field is present if the *Frame Type* 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). [25]

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 *Frame Type* sub-field is set to 0b01. 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 *GPDSrcID* field is not present if the *ApplicationID* sub-field of the *Extended NWK Frame Control* field is set to 0b001.

A.1.4.1.5 Endpoint field

The *Endpoint* field SHALL be present if *ApplicationID* = 0b010. It then carries the identifier of the GPD endpoint, which jointly with the GPD IEEE address identifies a unique logical GPD device.
If *ApplicationID* = 0b000 the *Endpoint* field SHALL be absent.

The values 0xf1 - 0xfe are reserved for future use. The value 0x00 indicates application endpoint-independent communication and SHOULD be used e.g. for channel and key updates. The value 0xff indicates 'all endpoints'.

[25] CCB #2146; Resolution added in 15-02014-005;

A.1.4.1.6 Security frame counter field

The presence and length of the *Security frame counter* field is dependent on the value of *ApplicationID* and *SecurityLevel* (see A.1.4.1.3).

A.1.4.1.7 GP Application Payload

If the *ApplicationID* sub-field of the *Extended NWK Frame Control* field is set to 0b000 or 0b010, the *GP application payload* is formatted as specified in Figure 9.

Octets: 1	**0/variable**
GPD CommandID	GPD Command payload
GP Application Payload	

Figure 9 – GP Application Payload for *ApplicationID* 0b000 and 0b010

The *CommandID* field carries the GP-specific command identifiers defined in the Green Power cluster (see Table 54 and Table 55). The *GPD command payload* field is a sequence of octets, and its presence and length is defined by the value of the *GPD CommandID* field.

A.1.4.1.8 MIC field

The *MIC* field carries the Message Integrity Code for this message, calculated as specified in sec. A.1.5.3.4. Its presence and length is dependent on the value of *ApplicationID* and *SecurityLevel* (see A.1.4.1.3).

A.1.4.2 Frame Types

A.1.4.2.1 Maintenance *Frame Type*

If the *Frame Type* 0b01 (Maintenance frame) is used, then the *GPD SrcID* field and the *Endpoint* field SHALL NOT be present. The GPD IEEE address in the MAC header SHOULD NOT be present. The security fields (*Security frame counter* and *MIC*) SHALL NOT be present and the frame SHALL be sent unprotected. If the GPDF is sent from the GPD, the *Extended NWK Frame Control* field SHALL be omitted. If the GPDF is sent to the GPD, the *Extended NWK Frame Control* field SHALL be omitted. In both cases, the *NWK Frame Control Extension* sub-field SHALL be set to 0b0.

A.1.4.2.2 Data *Frame Type*

The Data Frame Type SHALL be formatted as specified in sec. A.1.4.1.

A.1.5 Frame processing

A.1.5.1 cGP stub

Assuming the cGP-SAP, dGP-SAP and CZLP-SAP as described above, frames transmitted using the cGP stub are processed as described here.

A.1.5.1.1 GPDF reception

On receipt of a GPDF, the GP stub SHALL filter out (silently drop) frames with *ApplicationID* value other than 0b000, 0b010 and 0b001, frames with *Direction* sub-field of the *Extended NWK Frame Control* field set to 0b1, and duplicate frames.

Frames with *ApplicationID* 0b000 and 0b010 SHALL be passed up, using dGP-DATA.indication.

Frames with *ApplicationID* 0b001 SHALL be passed up, using dLPED-DATA.indication.

A.1.5.1.2 GPDF transmission

On reception of cGP-DATA.request from the dGP stub, the cGP stub constructs the GPDF with the *ApplicationID* sub-field of the *Extended NWK Frame Control* field set to 0b000 or 0b010, as supplied in the cGP-DATA.request primitive, and the remaining fields as supplied by the primitive.

On reception of dGP-DATA.request from the dLPED stub, the cGP stub constructs the GPDF with the *ApplicationID* sub-field of the *Extended NWK Frame Control* field set to 0b001 and the remaining fields as supplied by the primitive.

The constructed frame is then transmitted using MCPS-DATA.request.

Upon reception of the MCPS-DATA.confirm, the Status is passed on to dGP stub, using dGP-DATA.confirm.

A.1.5.2 dGP stub

Assuming the dGP-SAP, cGP-SAP and GP-SAP described above, frames transmitted using the dGP stub are processed as described here.

A.1.5.2.1 GPDF transmission

[26]On receipt of the GP-DATA.request primitive, the dGP stub SHALL check if the value of the SrcID parameter (in case of *ApplicationID* = 0b000) or GPD IEEE address parameter (in case of *ApplicationID* = 0b010) is from a valid range (see sec. A.1.4.1.4). If the check succeeds, the dGP stub SHALL then check the *gpTxQueue*.
If *ApplicationID* = 0b000, an entry with GPD SrcID identical to that in the received GPDF is sought for.
If *ApplicationID* = 0b010 an entry with GPD IEEE address identical to that in the received GPDF is sought for. Subsequently, the value of the *Tx on matching endpoint* sub-field of the *TxOptions* field of the queue entry and the GP-DATA.request and the *Endpoint* field of the *gpTxQueue* entry are analyzed. If the *Tx on matching endpoint* sub-field of the GP-DATA.request is set to 0b0, a suitable entry is found. If the Action parameter of the GP-DATA.request was set to TRUE, any additional gpTxQueue entries for the same IEEE address, if existent (if the *Tx on matching endpoint* sub-field in the found queue entry was set to 0b1) SHALL be removed and GP-DATA.confirm SHALL be returned with Status ENTRY_REMOVED. If the *Tx on the matching endpoint* sub-field of the GP-DATA.request is set to 0b1, AND either the *Tx on matching endpoint* sub-field of the analyzed entry is set to 0b0 or the *Tx on matching endpoint* sub-field of the analyzed entry is set to 0b1 and the value of the *Endpoint* field in the GP-DATA.request is equal to the value of the *Endpoint* field in the analyzed entry, a suitable entry is found.

If a suitable entry is found, and the Action parameter of the GP-DATA.request was set to FALSE, the previous GPDF is removed and GP-DATA.confirm with the Status ENTRY_REMOVED is provided to the Green Power EndPoint.
If a suitable entry is found, and the Action parameter of the GP-DATA.request was set to TRUE, the previous GPDF is overwritten and GP-DATA.confirm with the Status ENTRY_REPLACED is provided to the Green Power EndPoint.

If *ApplicationID* = 0b010, IEEE address matches, *Tx on matching endpoint* sub-field of both the GP-DATA.request and the analyzed entry are set to 0b1, but the value of the *Endpoint* fields differ, the analyzed entry SHALL NOT be removed. The dGP stub SHALL further search the gpTxQueue for an entry with identical IEEE address and identical Endpoint. If found, this entry SHALL be replaced by

[26] CCB #2360; Resolution added in 15-02014-011

the entry supplied in the GP-DATA.request and a GP-DATA.confirm with Status ENTRY_REPLACED is returned; if not found, the supplied entry SHALL be added to the queue.

If the *gpTxQueue* has no previous suitable entries for this GPD SrcID/GPD IEEE address and it has empty entries, the GPDF is added to the *gpTxQueue* and GP-DATA.confirm with the Status ENTRY_ADDED is provided to the Green Power EndPoint.

If the *gpTxQueue* has no previous suitable entries for this GPD SrcID/GPD IEEE address and it is full, the dGP stub returns GP-DATA.confirm with the Status set to QUEUE_FULL.

A.1.5.2.1.1 gpTxQueue

The gpTxQueue is a set of buffers for outgoing GPDF, implemented by a GP infrastructure device capable of bidirectional communication.

In gpTxQueue, GPDF are stored for transmission to GPD.

In its gpTxQueue, each GP infrastructure device SHALL have a maximum of only one pending GPDF frame per GPD SrcID or the combination of GPD IEEE address and Endpoint.

Each entry in the gpTxQueue SHALL have a gpTxQueueEntryLifetime parameter associated, initiated with the value in the GP-DATA.request with Action=TRUE. When this timeout elapses, the GP-DATA.confirm with the Status ENTRY_EXPIRED is returned to the Green Power EndPoint, the entry is cleared and can be used for any GPDF for any GPD ID.

A gpTxQueue of a GP Basic Proxy and Basic Sink/Basic Combo device SHALL have a minimum length of 1 entry. Since the basic devices do not support bidirectional communication in operation, the default entry lifetime is 0xffff (so that the entry will be cleared upon sending the GPDF or upon reception of GP-DATA.request with Action=FALSE). The basic devices are not required to be able to send secured GPDF.

For all other GP infrastructure device types the gpTxQueue SHALL have a minimum length of 5 entries.

A.1.5.2.1.2 gpTxOffset

The *gpTxOffset* is the time after which the GP stub SHALL send at least one GPDF in response to a GPDF with *RxAfterTx* sub-field set, if any present in the gpTxQueue for this GPD ID (and *Endpoint*, specific or 0xff, if *ApplicationID* = 0b010). It is measured on the medium, from the start of the reception of the first GPDF in a triggering GPFS, to the start of transmission of the first GPDF in the response GPFS.

The *gpTxOffset* has value identical to the *gpdRxOffset* (see sec. A.1.6.3.1).

If the GP stub misses a transmission window following a particular GPDF with *RxAfterTx* = 0b1 and defined by the *gpTxOffset* and *gpMaxTxOffsetVariation* parameters, it SHALL postpone the sending of the GPDF to the next transmission window.

The transmission time SHALL NOT exceed *gpTxDuration.*

A.1.5.2.1.3 gpMaxTxOffsetVariation

The *gpMaxTxOffsetVariation* is the maximum allowed deviation to the gpTxOffset, as measured on the medium.

The *gpMaxTxOffsetVariation* has the non-negative value of 5ms.

Thus, the GP stub SHALL commence the transmission of a response GPDF not earlier than 20ms and not later than 25ms from the start of the reception of the triggering GPFS.

A.1.5.2.1.4 gpTxDuration

The *gpTxDuration* is the maximum allowed transmission time for the GP stub. Thus, depending on the GPDF length, the GP stub MAY send the GPDF more than once, to increase the reliability of communication, taking into consideration that the *gpdMinRxWindow* of the receiving GPD may be shorter than the *gpTxDuration*. It is measured on the medium from the start of the transmission of the first GPDF in a given GPFS, to the end of the last GPDF in a given GPFS.

The *gpTxDuration* has the value of 10ms.

A.1.5.2.2 GPDF reception

On receipt of a dGP-DATA.indication, the dGP stub SHALL proceed as follows.

[27]If the received frame was of type Maintenance frame (0b01), and the *GPD CommandID* of the received GPDF does NOT have a value from the range 0xf0-0xff, then the dGP stub SHALL schedule transmission of the GPDF for *ApplicationID* = 0b000, SrcID = 0x00000000 stored in the gpTxQueue, if any, with *UseCSMA* parameter set to FALSE, so that between *gpTxOffset* and *gpTxOffset* + *gpMaxTxOffsetVariation* after reception of the triggering GPDF (as measured on the medium) at least one GPDF is sent by the dGP stub; to that end, the dGP stub will send a CGP-DATA.request; the transmission time by the dGP stub SHALL NOT exceed *gpTxDuration*; MAC acknowledgement SHALL NOT be requested. On reception of the dGP-DATA.confirm, the dGP calls GP-DATA.confirm with Status value copied from the dGP-DATA.confirm; if the Status in the dGP-DATA.confirm is SUCCESS, it removes this gpTxQueue entry. Subsequently, the dGP stub indicates reception of the GPDF to the next higher layer, by calling GP-DATA.indication; since in the current version of the specification security is not used for Maintenance frames (*Frame Type* = 0b01), the dGP calls GP-DATA.indication with the Status NO_SECURITY.
If the received *GPD CommandID* had a value from the range 0xf0-0xff, the dGP SHALL silently drop it.

If the received frame was of type Data frame (0b00) the dGP stub SHALL proceed as follows.

[28]The dGP stub SHALL check if the value of the SrcID parameter (in case of *ApplicationID* = 0b000) or GPD IEEE address parameter (in case of *ApplicationID* = 0b010) is from a valid range (see sec. A.1.4.1.4). If the check succeeds, the dGP stub SHALL check the *SecurityLevel*. If the *SecurityLevel* is not supported (incl. *SecurityLevel* = 0b01), the dGP stub SHALL silently drop the frame. If *SecurityLevel* is supported and has the value of 0b00 or 0b10, and *GPD CommandID* has the value from the range 0xf0-0xff, the GPDF is silently dropped. If *SecurityLevel* is supported, the dGP stub then generates GP-SEC.request and waits for GP-SEC.response.

On receipt of GP-SEC.response with *Status* DROP_FRAME, the dGP stub drops the frame. On receipt of GP-SEC.response with Status PASS_UNPROCESSED, the dGP stub generates GP-DATA.indication for the unprocessed frame, [29]with *Status* UNPROCESSED. On receipt of GP-SEC.response with Status MATCH or TX_THEN_DROP, the dGP stub security-processes the received GPDF, as described in A.1.5.3.5.

If security processing fails, the dGP stub indicates that with GP-DATA.indication carrying the corresponding *Status* value and stops any further processing of this frame.

[27] CCB #2135; Resolution added in 15-02014-003; Resolution modified in 15-02014-004: Moved the MAC ACK requirement from A.3.9.1 to here.
[28] CCB #2360; Resolution added in 15-02014-011
[29] CCB #2362; Resolution added in 15-02014-011

If security processing is successful, and the *SecurityLevel* was 0b11, the dGP stub checks the plaintext value of the *GPD CommandID*. If it has the value from the range 0xf0-0xff, the GPDF is silently dropped.

If security processing was successful, and the GPD CommandID is not from the 0xf0 – 0xff range, the dGP stub checks if the *RxAfterTx* sub-field of the *Extended NWK Frame Control* field of the received GPDF was set to 0b1. If yes, it searches the *gpTxQueue* for an entry. If *ApplicationID* = 0b000, an entry with GPD SrcID identical to that in the received GPDF is sought for. If *ApplicationID* = 0b010 an entry with GPD IEEE address identical to that in the received GPDF is sought for. Subsequently, the value of the *Tx on matching endpoint* sub-field of the *TxOptions* field and the *Endpoint* field of the *gpTxQueue* entry is analyzed. If the *Tx on matching endpoint* sub-field set to 0b0, the *Endpoint* field is ignored, and a suitable GPDF is found. If the *Tx on matching endpoint* sub-field set to 0b1, and the value of the *Endpoint* field of the *gpTxQueue* entry is identical to that in the received GPDF, a suitable GPDF is found. If a suitable GPDF is found, dGP stub triggers security processing of the to-be-sent GPDF with the same security input parameters as for the received GPDF. If the Data *Frame Type* is used, the *NWK Frame Control Extension* sub-field SHALL be set to 0b1, the *Extended NWK Frame Control* field SHALL be present, and the *RxAfterTx* sub-field SHALL be set to 0b0 and the *Direction* sub-field SHALL be set to 0b1. Then, the dGP stub schedules GPDF transmission by sending CGP-DATA.request, with *UseCSMA* parameter set to FALSE, so that between *gpTxOffset* and *gpTxOffset* + *gpMaxTxOffsetVariation* after reception of the triggering GPDF (as measured on the medium) at least one GPDF is sent by the dGP stub; the transmission time by the dGP stub SHALL NOT exceed *gpTxDuration*. On reception of the dGP-DATA.confirm, the dGP calls GP-DATA.confirm with Status value copied from the dGP-DATA.confirm; if the Status in the dGP-DATA.confirm is SUCCESS, it removes this gpTxQueue entry. Then, if the *Status* of the GP-SEC.response was TX_THEN_DROP, the dGP silently drops the received GPDF.

Otherwise, if the Status of the GP-SEC.response was MATCH, and if no matching entry is found in the *gpTxQueue*, the GP stub indicates reception of the GPDF to the next higher layer, by calling GP-DATA.indication. If *SecurityLevel* was 0b00, the dGP calls GP-DATA.indication with the Status NO_SECURITY; if *SecurityLevel* was 0b10 – 0b11, the dGP calls GP-DATA.indication with the Status SECURITY_SUCCESS.

A.1.5.3 Security operation of the GP stub

A.1.5.3.1 Per GPDF Security Level and Key selection

The dGP stub SHALL:

- For the incoming secured GPDF: use the parameters supplied by the GP-SEC.response.
- For the outgoing secured GPDF: use the same key and protection level as for the triggering GPDF.

A.1.5.3.2 Constructing AES Nonce

The AES nonce, defined by the Zigbee specification (sec. 4.5.2.2 of [1]) to have the format as depicted in Figure 10, is used for security operations and SHALL be constructed in the following way.

Octets: 8	4	1
Source address	Frame counter	Security control

Figure 10 – Format of the AES nonce [1]

For *ApplicationID* = 0b000, the *Source address* parameter SHALL take the value:

- for the incoming secured GPDF (i.e. the GPDF sent by the GPD): SourceAddress[63:32] = SrcID,

SourceAddress[31:0] = SrcID;
- for the outgoing secured GPDF (i.e. the GPDF sent to the GPD): SourceAddress[63:32] = SrcID, SourceAddress[31:0] = 0;

where the SrcID is little Endian (LSB first).
For example, if the SrcID = 0x87654321, the *Source address* parameter takes the following values:

- for the incoming secured GPDF: 0x8765432187654321 = { 0x21, 0x043, 0x65, 0x87, 0x21, 0x43, 0x65, 0x87 };
- for the outgoing secured GPDF: 0x8765432100000000 = { 0x00, 0x00, 0x00, 0x00, 0x21, 0x43, 0x65, 0x87 }.

For *ApplicationID* = 0b010, the *Source address* parameter SHALL take the value of the IEEE address of the GPD, for both incoming and outgoing secured GPDF.

Note: the *Endpoint* field, which is mandatory in case of *ApplicationID* = 0b010 is NOT used for nonce generation; it is only part of the GPDF's authenticated header.

Frame counter parameter SHALL take the value:

- for the incoming secured GPDF: 4B frame counter for this GPD, as transmitted in the GPDF;
- for the outgoing secured GPDF: the 4B value of frame counter that was last used by this GPD (i.e. the frame counter value from the GPDF received from this GPD with *RxAfterTx*=TRUE that immediately precedes the sending of this frame to the GPD).

Security control field, defined to be part of the AES nonce by the Zigbee specification [1] and formatted as shown in Figure 11, is never exchanged between the GP devices. Thus, for interoperability, the values used SHALL be as defined below.

Bit: 0-2	3-4	5	6-7
Security level	Key identifier	Extended nonce	Reserved

Figure 11 – Format of the Security Control field of the AES Nonce [1]

- Security level (according to [1])= 0b101
- Key identifier (NOT according to [1]) = 0b00
- Note that this security level and Key identifier are never transmitted and are NOT used for determining the transformation applied to the packet, since those are governed by the *Security* sub-field of the NWK Frame Control field of the GPDF. The values here are defined for interoperability only.
- Extended nonce =0b0;
- Reserved =
  - For *ApplicationID* = 0b000 and for incoming secured GPDF (i.e. GPDF sent by GPD): *Reserved* = 0b00;
  - For outgoing secured GPDF (i.e. GPDF sent to GPD) with an *ApplicationID* = 0b010: *Reserved* = 0b11.

The *Nonce* SHALL be formatted little endian, i.e. LSB first. Also the fields *Source address* and *Frame counter* SHALL be little endian, i.e. LSB first.

A.1.5.3.3 Initialization

If the *SecurityLevel* field of the GPDF has the value 0b10 or 0b11, the following transformation applies.

The definition *Payload* is applied to the following fields of the GPDF:

Payload = GPD CommandID || GPD Command Payload.

[30]The definition *Header* is applied to the following fields of the GPDF:

in case of *ApplicationID* = 0b000:

Header = NWK Frame Control || Ext NWK Frame Control || SrcID || Frame counter;

in case of *ApplicationID* = 0b010:

Header = NWK Frame Control || Ext NWK Frame Control || Endpoint || Frame counter.

A.1.5.3.4 Outgoing frames encryption and authentication

Determine the security level, as described in A.1.5.2.2, and perform initialization, as described in A.1.5.3.3.

A.1.5.3.4.1 CCM* execution

Execute the CCM* mode encryption and authentication operation, as specified in Annex A of [1]. The following parameters are used:

- The parameter M is =4, which means that 4B MIC is calculated (irrespective of *gpdSecurityLevel*).
- Nonce is constructed as described in A.1.5.3.2.
- The bit string *Key* determined as described in A.1.5.2.2.
- if the frame requires encryption (as indicated by *gpdSecurityLevel* = 0b11),
  - the octet string *a* SHALL be the *Header*, as defined in A.1.5.3.3,
  - and the octet string *m* SHALL be the string *Payload*, as defined in A.1.5.3.3,
- Otherwise, [31]if the frame does not use encryption (as indicated by the *gpdSecurityLevel* parameter equal to 0b10),
  - the octet string *a* SHALL be the string *Header* || *Payload*, as defined in A.1.5.3.3,
  - and the octet string *m* SHALL be a string of length zero.

The output CCM* is the string *c*, which consists of right-concatenation of the encrypted message *Ciphertext* and the encrypted authentication tag *U*.

A.1.5.3.4.2 Constructing protected GPDF

For transmission of the protected GPDF:

- Else, if the security level, as indicated by *gpdSecurityLevel* = 0b10:
  - The fields *GPD CommandID* and *GPD Command Payload* remain unmodified;
  - 4 LSB of *U* are inserted into GPDF *MIC* field.
  - The *Frame counter* used for frame protection is inserted into GPDF *Security frame counter* field.
- Else if the security level, as indicated by the *gpdSecurityLevel* = 0b11:
  - The *Ciphertext* is used as *Payload*, i.e. the *Ciphertext* replaces the fields *GPD CommandID* and *GPD Command payload*;
  - 4 LSB of *U* are inserted into GPDF *MIC* field;
  - The *Frame counter* used for frame protection is inserted into GPDF *Security frame counter* field.

[30] CCB #2345; Resolution added in 15-02014-011
[31] CCB #2431; resolution added in 15-02014-010

A.1.5.3.5 Incoming frames decryption and authentication check

Determine the security level, as described in A.1.5.2.2, and perform initialization, as described in A.1.5.3.3.

The following parameters are used for CCM* mode encryption and authentication operation, as specified in Annex A of [1]:

- The parameter M is =4.
- Nonce is constructed as described in A.1.5.3.2.
- The bit string *Key* determined as described in A.1.5.2.2.

If decryption is required (*SecurityLevel* 0b11), proceed with CCM* as specified in A.2.3 of [1], by using *PlaintextData* = encrypted GPD CommandID || encrypted GPD Command Payload from the received GPDF.

For authentication (for all *SecurityLevel* 0b10 - 0b11), calculate the *U*, as defined in A.1.5.3.4.1, taking the decrypted *GPD CommandID* and *GPD Command Payload* fields as *Payload*, and the *Header* fields as defined in A.1.5.3.3. Subsequently, compare the *MIC* field of the received GPDF with the corresponding number of LSB of the calculated *U*.

Subsequently, the results are evaluated as described in A.1.5.3.5.1.

A.1.5.3.5.1 Reporting to next higher layer

If the authentication is successful, dGP stub calls GP-DATA.indication with Status SECURITY_SUCCESS and carrying the unprotected GPD CommandID and GPD Command Payload.

If the authentication is not successful, and *SecurityLevel*=0b10 or 0b11, dGP stub calls GP-DATA.indication with Status AUTH_FAILED and carrying the protected GPD CommandID and GPD Command Payload.

A.1.5.4 Security test vectors for ApplicationID = 0b000 and a shared key

The parameters underlined are dependent on device application and capabilities and thus could have other values.

A.1.5.4.1 Common settings

- GP Security Key = [0xC0 , 0xC1 , 0xC2 , 0xC3 , 0xC4 , 0xC5 , 0xC6 , 0xC7 , 0xC8 , 0xC9 , 0xCa , 0xCb , 0xCc , 0xCd , 0xCe , 0xCf] = 0xCFCECDCCCBCAC9C8C7C6C5C4C3C2C1C0
- MAC fields:
  - Dest PANId = 0xffff
  - Dest Addr = 0xffff
  - MAC SeqNum = 0x02
- NWK fields:
  - NWK FC := [Ext NWK Header = 0b1 || Auto-Commissioning =0b0|| Zigbee Protocol 0b0011 || Frame type =0b00] → [0b10001100] 0x8c
  - GPD SrcID = 0x87654321
  - Security Frame Counter = 0x00000002
- Application fields:
  - GPD CommandID = 0x20 (OFF)

- No data payload

A.1.5.4.2 SecurityLevel=0b10

A.1.5.4.2.1 Transmitted packet

Transmitted packet = MAC FC || MAC header || GP stub NWK header || Payload || MIC

Transmitted packet
18 01 08 02 FF FF FF FF 8C 10 21 43 65 87 02 00 00 00 20 CF 78 7E 72

A.1.5.4.2.2 Inputs

- NWK fields:
  - NWK FC Extended = [Direction = 0b0 || RxAfterTx = 0b0 || SecurityKey = 0b0 ||SecurityLevel = 0b10 || ApplicationID = 0b000] →0b00010000 → 0x10

A.1.5.4.2.3 GP Security Calculation

Definitions
- Nonce N = [0x21, 0x43, 0x65, 0x87, 0x21, 0x43, 0x65, 0x87, 0x02, 0x00, 0x00, 0x00, 0x05]

a = header || Payload

Header = NWK FC || NWK_EXT FC || SrcID || Security Frame Counter.
header = 0x8c || 0x10 || 0x87654321 || 0x00000002
header = [0x8c, 0x10, 0x21, 0x43, 0x65, 0x87, 0x02, 0x00, 0x00, 0x00]

payload = 0x20

a = 0x8c || 0x10 || 0x87654321 || 0x00000002 || 0x20
a = [0x8c, 0x10, 0x21, 0x43, 0x65, 0x87, 0x02, 0x00, 0x00, 0x00; 0x20]

Calculation
l(a) = 0x0b
L(a) = 0x00 0x0b

AddAuthData = L(a) || a || padding
AddAuthData = [0x00, 0x0b, 0x8c, 0x10, 0x21, 0x43, 0x65, 0x87, 0x02, 0x00, 0x00, 0x00, 0x20, 0x00, 0x00, 0x00]

Flags = [Reserved = 0b0 ||Adata = 0b1 || (M-2)/2 = 0b001 || (L-1) = 0b001 → 0x49]

B0 = [Flags =0x49|| Nonce N = 0x21 0x43 0x65 0x87 0x21 0x43 0x65 0x87, 0x02, 0x00, 0x00, 0x00, 0x05 || 0x00 0x00]

Result
U = **0x727E78CF**
MIC = FULL U = 0x727E78CF = [0xCF, 0x78, 0x7E, 0x72]

A.1.5.4.3 SecurityLevel=0b11

A.1.5.4.3.1 Transmitted packet

Transmitted packet = MAC FC || header || Payload || MIC

Transmitted packet
18 01 08 02 FF FF FF FF 8C 18 21 43 65 87 02 00 00 00 83 CA 43 24 DD

A.1.5.4.3.2 Inputs

- NWK fields:
  - NWK FC Extended = [Direction = 0b0 || RxAfterTx = 0b0 || SecurityKey = 0b0 ||SecurityLevel = 0b11 || ApplID = 0b000] →0b00011000 → 0x18

A.1.5.4.3.3 GP Security Calculation

<u>Definitions</u>
- Nonce N = [0x21, 0x43, 0x65, 0x87, 0x21, 0x43, 0x65, 0x87, 0x02, 0x00, 0x00, 0x00, 0x05]

a = Header
m = Payload

Header = NWK FC || NWK_EXT FC || SrcID || Security Frame Counter.
header = 0x8c || 0x18 || 0x87654321 || 0x00000002
header = [0x8c, 0x18, 0x21, 0x43, 0x65, 0x87, 0x02, 0x00, 0x00, 0x00]

payload = 0x20

a = 0x8c || 0x18 || 0x87654321 || 0x00000002
a = [0x8c, 0x18, 0x21, 0x43, 0x65, 0x87, 0x02, 0x00, 0x00, 0x00]

m = 0x20

<u>Calculation</u>
l(a) = 0x0a
L(a) = 0x00 0x0a

AddAuthData = L(a) || a || padding
AddAuthData = [0x00, 0x0a, 0x8c, 0x18, 0x21, 0x43, 0x65, 0x87, 0x02, 0x00, 0x00, 0x00, 0x20, 0x00, 0x00, 0x00]

PlaintextData = m || padding
PlaintextData = [0x20, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00]

AuthData = AddAuthData || PlaintextData
AuthData = [0x00, 0x0a, 0x8c, 0x18, 0x21, 0x43, 0x65, 0x87, 0x02, 0x00, 0x00, 0x00, 0x20, 0x00, 0x00, 0x00, 0x20, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00]

FlagsAuth = [Reserved = 0b0 ||Adata = 0b1 || (M-2)/2 = 0b001 || (L-1) = 0b001 → 0x49]

B0 = [FlagsAuth =0x49|| Nonce N = 0x21 0x43 0x65 0x87 0x21 0x43 0x65 0x87, 0x02, 0x00, 0x00, 0x00, 0x05 || l(m) = 0x00 0x01]

B1 = [0x00, 0x0a, 0x8c, 0x18, 0x21, 0x43, 0x65, 0x87, 0x02, 0x00, 0x00, 0x00, 0x20, 0x00, 0x00, 0x00]

B2 = [0x20, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00]

FlagsEncrypt = [Reserved = 0b0 ||[Reserved = 0b0 || 0b000 || (L-1) = 0b001 → 0x01]

Ai = [FlagsEncrypt = 0x01 || Nonce N = 0x21 0x43 0x65 0x87 0x21 0x43 0x65 0x87, 0x02, 0x00, 0x00, 0x00, 0x05 || Counter = 0x00 0x0i]

Result
U = **0xDD2443CA**
MIC = FULL U = 0xDD2443CA = [0xCA, 0x43, 0x24, 0xDD]

Cipher = **0x83**

A.1.5.5 Security test vectors for ApplicationID = 0b000 and an individual key

A.1.5.5.1 Common settings

- GP Security Key = [0xC0 , 0xC1 , 0xC2 , 0xC3 , 0xC4 , 0xC5 , 0xC6 , 0xC7 , 0xC8 , 0xC9 , 0xCa , 0xCb , 0xCc , 0xCd , 0xCe , 0xCf] = 0xCFCECDCCCBCAC9C8C7C6C5C4C3C2C1C0
- Nonce = 21 43 65 87 21 43 65 87 02 00 00 00 05
- MAC fields:
  - Dest PANId = 0xffff
  - Dest Addr = 0xffff
  - MAC SeqNum = 0x02
- NWK fields:
  - NWK FC := [Ext NWK Header = 0b1 || Auto-Commissioning =0b0|| Zigbee Protocol 0b0011 || Frame type =0b00] → [0b10001100] 0x8c
  - GPD SrcID = 0x87654321
  - Security Frame Counter = 0x00000002
- Application fields:
  - GPD CommandID = 0x20 (OFF)
  - No data payload

A.1.5.5.2 SecurityLevel=0b10

Extended NWK FC = [Direction = 0b0 || RxAfterTx = 0b0 || SecurityKey = 0b1 || SecurityLevel = 0b10 || ApplID = 0b000] →0x30

Over the air packet:

18 01 08 02 FF FF FF FF 8C 30 21 43 65 87 02 00 00 00 20 AD 69 A9 78

A.1.5.5.3 SecurityLevel=0b11

Extended NWK FC = [Direction = 0b0 || RxAfterTx = 0b0 || SecurityKey = 0b1 || SecurityLevel = 0b11 || ApplID = 0b000] →0x38

Over the air packet:

```
18 01 08 02 FF FF FF FF 8C 38 21 43 65 87 02 00 00 00 83 5F 1A 30 34
```

zigbee alliance

A.1.5.6 Security test vectors for ApplicationID = 0b000 and bidirectional operation

A.1.5.6.1 Common settings

For all frames

- NWK *Frame Type* sub-field = 0b00
- *Zigbee Protocol Version* sub-field = 0b0011
- *Auto-Commissioning* sub-field = 0b0
- *NWK Frame Control Extension* sub-field = 0b1
- GPD SrcID = 0x87654321
- Security Frame Counter = 0x44332211
- Security Key = { 0xC0 0xC1 0xC2 0xC3 0xC4 0xC5 0xC6 0xC7 0xC8 0xC9 0xCA 0xCB 0xCC 0xCD 0xCE 0xCF }

For incoming frames (from GPD to GPP / GPS)

- *RxAfterTx* sub-field = 0b1
- *Direction* sub-field = 0b0
- MAC Seq Nbr
  - For SecurityLevel = 0b10 or 0b11: 0x01
- GPD CommandID = 0x20 (OFF)
- GPD Command payload = ∅ (No payload)

For outgoing frames (from GPP/GPS to GPD)

- *RxAfterTx* sub-field = 0b0
- *Direction* sub-field = 0b1
- MAC Seq Nbr = 39
- GPD CommandID = 0xF3 (Channel Configuration)
- GPD Command payload = 0x00 (channel 11, bidirectional GPS)

A.1.5.6.2 Security test vectors for a shared key

For all test vectors with a shared security key:

- *SecurityKey* sub-field of *Extended NWK Frame Control* field = 0b0 (shared key)

A.1.5.6.2.1 SecurityLevel = 0b10

Incoming frame (GPD to GPP / GPS)

0x18 0x01 0x08 0x01 0xFF 0xFF 0xFF 0xFF 0x8C 0x50 0x21 0x43 0x65 0x87 0x11 0x22 0x33 0x44 0x20 **0xF6 0x36 0x78 0x9E**

Full 4B MIC: 0x**9E7836F6**

Outgoing frame (GPP/GPS to GPD)

0x19 0x01 0x08 0x39 0xFF 0xFF 0xFF 0xFF 0x8C 0x90 0x21 0x43 0x65 0x87 0x11 0x22 0x33 0x44 0xF3 0x00 **0xCC 0xA0 0xBB 0x2E**

Full 4B MIC: 0x**2EBBA0CC**

A.1.5.6.2.2 SecurityLevel = 0b11

Incoming frame (GPD to GPP / GPS)

0x18 0x01 0x08 0x01 0xFF 0xFF 0xFF 0xFF 0x8C 0x58 0x21 0x43 0x65 0x87 0x11 0x22 0x33 0x44 0x2A **0x3D 0x17 0x0A 0xAA**

Encrypted data: <u>0x2A</u>

Full 4B MIC: 0x**AA0A173D**

<u>**Outgoing frame (GPP/GPS to GPD)**</u>

0x19 0x01 0x08 0x39 0xFF 0xFF 0xFF 0xFF 0x8C 0x98 0x21 0x43 0x65 0x87 0x11 0x22 0x33 0x44 <u>0x9E 0x7E</u> **0x14 0x0F 0xB5 0xDA**

Encrypted data: <u>0x9E 0x7E</u>

Full 4B MIC: 0x**DAB50F14**

A.1.5.6.3 Security test vectors for an individual key

For all test vectors with an individual key:

- *SecurityKey* sub-field in *Extended NWK Frame Control* field = 0b1 (individual key)

A.1.5.6.3.1 SecurityLevel = 0b10

<u>**Incoming frame (GPD to GPP / GPS)**</u>

0x18 0x01 0x08 0x01 0xFF 0xFF 0xFF 0xFF 0x8C 0x70 0x21 0x43 0x65 0x87 0x11 0x22 0x33 0x44 0x20 **0x6E 0xA9 0x51 0xBC**

Full 4B MIC: 0x**BC51A96E**

<u>**Outgoing frame (GPP/GPS to GPD)**</u>

0x19 0x01 0x08 0x39 0xFF 0xFF 0xFF 0xFF 0x8C 0xB0 0x21 0x43 0x65 0x87 0x11 0x22 0x33 0x44 0xF3 0x00 **0xF9 0xF1 0x7C 0x8A**

Full 4B MIC: 0x**8A7CF1F9**

A.1.5.6.3.2 SecurityLevel = 0b11

<u>**Incoming frame (GPD to GPP / GPS)**</u>

0x18 0x01 0x08 0x01 0xFF 0xFF 0xFF 0xFF 0x8C 0x78 0x21 0x43 0x65 0x87 0x11 0x22 0x33 0x44 <u>0x2A</u> 0x**D9 0xF0 0x08 0x6D**

Encrypted data: <u>0x2A</u>

Full 4B MIC: 0x**6D08F0D9**

<u>**Outgoing frame (GPP/GPS to GPD)**</u>

0x19 0x01 0x08 0x39 0xFF 0xFF 0xFF 0xFF 0x8C 0xB8 0x21 0x43 0x65 0x87 0x11 0x22 0x33 0x44 0x9E 0x7E 0xD6 0x6E 0x60 0x08

Encrypted data: <u>0x9E 0x7E</u>

Full 4B MIC: 0x**08606ED6**

A.1.5.7 Security test vectors for key derivation

A.1.5.7.1 NWK-key derived GPD group key

Input:

Zigbee NWK key = {0x01, 0x03, 0x05, 0x07, 0x09, 0x0b, 0x0d, 0x0f, 0x00, 0x02, 0x04, 0x06, 0x08, 0x0a, 0x0c, 0x0d};

Output:

NWK-key derived GPD group key = {0xBA, 0x88, 0x86, 0x7f, 0xc0, 0x09, 0x39, 0x87, 0xeb, 0x88, 0x64, 0xce, 0xbe, 0x5f, 0xc6, 0x13};

A.1.5.7.2 Derived individual GPD key

Input:

SrcID = 0x87654321;

GPD Group Key = {0xc0, 0xc1, 0xc2, 0xc3, 0xc4, 0xc5, 0xc6, 0xc7, 0xc8, 0xc9, 0xca, 0xcb, 0xcc, 0xcd, 0xce, 0xcf};

Output:

Derived individual GPD key = {0x7a, 0x3a, 0x73, 0x43, 0x8d, 0x6e, 0x47, 0x55, 0x28, 0x81, 0xa0, 0x28, 0xad, 0x59, 0x23, 0x2e};

A.1.5.8 Security test vectors for TC-LK protection

A.1.5.8.1 OOB key in Commissioning GPDF for SrcID=0x12345678

Input:

SrcID = 0x12345678

OOB Key = {0xC0 0xC1 0xC2 0xC3 0xC4 0xC5 0xC6 0xC7 0xC8 0xC9 0xCA 0xCB 0xCC 0xCD 0xCE 0xCF}

TC-LK = {0x5A 0x69 0x67 0x42 0x65 0x65 0x41 0x6C 0x6C 0x69 0x61 0x6E 0x63 0x65 0x30 0x39}

Security frame counter – irrelevant;

Calculation:

Nonce = {0x78 0x56 0x34 0x12 0x78 0x56 0x34 0x12 0x78 0x56 0x34 0x12 0x05}

Header = {0x78 0x56 0x34 0x12}

Plaintext = {0xC0 0xC1 0xC2 0xC3 0xC4 0xC5 0xC6 0xC7 0xC8 0xC9 0xCA 0xCB 0xCC 0xCD 0xCE 0xCF}

Output:

TC-LK protected OOB key = {0x7D 0x17 0x7B 0xD2 0x9E 0xA0 0xFD 0xA6 0xB0 0x17 0x03 0x65 0x87 0xDC 0x26 0x00}

GPDkeyMIC = {0x61 0xF1 0x63 0xA9}

A.1.5.8.2 Another OOB key in Commissioning GPDF for SrcID=0x12345678

Input:

SrcID = 0x12345678

OOB Key = {0x16 0x68 0x16 0x68 0x16 0x68 0x16 0x68 0x16 0x68 0x16 0x68 0x16 0x68 0x16 0x68}

TC-LK ={0x5A 0x69 0x67 0x42 0x65 0x65 0x41 0x6C 0x6C 0x69 0x61 0x6E 0x63 0x65 0x30 0x39}

Security frame counter – irrelevant;

Calculation:

Nonce = {0x78 0x56 0x34 0x12 0x78 0x56 0x34 0x12 0x78 0x56 0x34 0x12 0x05}

Header = {0x78 0x56 0x34 0x12}

Plaintext = {0x16 0x68 0x16 0x68 0x16 0x68 0x16 0x68 0x16 0x68 0x16 0x68 0x16 0x68 0x16 0x68}

Output:

TC-LK protected OOB key = {0xAB 0xBE 0xAF 0x79 0x4C 0x0D 0x2D 0x09 0x6E 0xB6 0xDF 0xC6 0x5D 0x79 0xFE 0xA7}

GPDkeyMIC = {0x67 0x31 0x42 0x6A}

A.1.5.8.3 Shared key in Commissioning Reply GPDF for SrcID=0x12345678

Input:

SrcID = 0x12345678

Shared Key = {0xC0 0xC1 0xC2 0xC3 0xC4 0xC5 0xC6 0xC7 0xC8 0xC9 0xCA 0xCB 0xCC 0xCD 0xCE 0xCF}

TC-LK = {0x5A 0x69 0x67 0x42 0x65 0x65 0x41 0x6C 0x6C 0x69 0x61 0x6E 0x63 0x65 0x30 0x39}

Security frame counter from the GPDF that triggers Commissioning Reply *creation*, not *sending* = 3;

Calculation:

Nonce = {0x00 0x00 0x00 0x00 0x78 0x56 0x34 0x12 0x04 0x00 0x00 0x00 0x05}

Header = {0x78 0x56 0x34 0x12}

Plaintext = {0xC0 0xC1 0xC2 0xC3 0xC4 0xC5 0xC6 0xC7 0xC8 0xC9 0xCA 0xCB 0xCC 0xCD 0xCE 0xCF}

Output:

TC-LK protected shared key = {0xE9 0x00 0x06 0x63 0x1D 0x0D 0xFD 0xC6 0x38 0x06 0x8E 0x5E 0x69 0x67 0xD3 0x25}

GPDkeyMIC = {0x27 0x55 0x9F 0x75}

Frame Counter = {0x04 0x00 0x00 0x00}

A.1.5.9 Security test vectors for *ApplicationID* = 0b010 and a shared key; *Direction* = 0b0 (from GPD)

The parameters marked with violet are dependent on device application and capabilities and thus could have other values.

A.1.5.9.1 Common settings

- GP Security Key = [0xC0 , 0xC1 , 0xC2 , 0xC3 , 0xC4 , 0xC5 , 0xC6 , 0xC7 , 0xC8 , 0xC9 , 0xCa , 0xCb , 0xCc , 0xCd , 0xCe , 0xCf] = 0xCFCECDCCCBCAC9C8C7C6C5C4C3C2C1C0
- GPD IEEE address = 0x8877665544332211
- Endpoint = 0x0A
- MAC fields:
  - Dest PANId = 0xffff
  - MAC SeqNum = 0x02
- NWK fields:
  - NWK FC := [Ext NWK Header = 0b1 || Auto-Commissioning =0b0 || Zigbee Protocol 0b0011 || Frame type =0b00] → [0b10001100] 0x8c
  - Security Frame Counter = 0x00000002
- Application fields:
  - GPD CommandID = 0x20 (OFF)
  - No data payload

A.1.5.9.2 [32]SecurityLevel=0b10

A.1.5.9.2.1 Transmitted packet

[32] CCB #2346; Resolution added in 15-02014-011

Transmitted packet = MAC FC || MAC header || GP stub NWK header || Payload || MIC
Transmitted packet
12 41 C8 02 FF FF FF FF 11 22 33 44 55 66 77 88 8C 12 0A 02 00 00 00 20 C5 A8 3C 5E

A.1.5.9.2.2 Inputs

Extended NWK FC = [Direction = 0b0 || RxAfterTx = 0b0 || SecurityKey = 0b0 || SecurityLevel = 0b10 || ApplID = 0b010] →0x12
SrcID field: absent;

A.1.5.9.2.3 GP Security Calculation

Definitions
Nonce N = [0x11, 0x22, 0x33, 0x44, 0x55, 0x66, 0x77, 0x88, 0x02, 0x00, 0x00, 0x00, 0x05]

a = header || Payload

Header = NWK FC || NWK_EXT FC || Endpoint || Security Frame Counter.
header = 0x8c || 0x12 || 0x0A || 0x00000002
header = [0x8c, 0x12, 0x0A, 0x02, 0x00, 0x00, 0x00]

payload = 0x20

a = 0x8c || 0x12 || 0x0A || 0x00000002 || 0x20
a = [0x8c, 0x12, 0x0A, 0x02, 0x00, 0x00, 0x00; 0x20]

Calculation
l(a) = 0x08
L(a) = 0x00 0x08

AddAuthData = L(a) || a || padding
AddAuthData = [0x00, 0x08, 0x8c, 0x12, 0x0A, 0x02, 0x00, 0x00, 0x00, 0x20, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00]

Flags = [Reserved = 0b0 ||Adata = 0b1 || (M-2)/2 = 0b001 || (L-1) = 0b001 → 0x49]

B0 = [Flags =0x49|| Nonce N = 0x11 0x22 0x33 0x44 0x55 0x66 0x77 0x88, 0x02, 0x00, 0x00, 0x00, 0x05 || 0x00 0x00]

Result
U = **0x5E3CA8C5**
MIC = FULL U = 0x5E3CA8C5 = [0xC5, 0xA8, 0x3C, 0x5E]

A.1.5.9.3 [33]SecurityLevel=0b11

A.1.5.9.3.1 Transmitted packet

Transmitted packet = MAC FC || MAC header || GP stub NWK header || Payload || MIC
Transmitted packet

[33] CCB #2346; Resolution added in 15-02014-011

12 41 C8 02 FF FF FF FF 11 22 33 44 55 66 77 88 8C 1A 0A 02 00 00 00 7E D2 A2 36 1B

A.1.5.9.3.2 Inputs

Extended NWK FC = [Direction = 0b0 || RxAfterTx = 0b0 || SecurityKey = 0b0 || SecurityLevel = 0b11 || ApplID = 0b010] →0x1A

SrcID field: absent;

A.1.5.9.3.3 Security Calculation

Definitions

Nonce N = [0x11, 0x22, 0x33, 0x44, 0x55, 0x66, 0x77, 0x88, 0x02, 0x00, 0x00, 0x00, 0x05]

a = Header
m = Payload

Header = NWK FC || NWK_EXT FC || Endpoint || Security Frame Counter
header = 0x8C || 0x1A || 0x0A || 0x00000002
header = [0x8C, 0x1A, 0x0A, 0x02, 0x00, 0x00, 0x00]

payload = 0x20

a = 0x8C || 0x1A || 0x0A || 0x00000002
a = [0x8C, 0x1A, 0x0A, 0x02, 0x00, 0x00, 0x00]

m = 0x20

Calculation
l(a) = 0x07
L(a) = 0x00 0x07

AddAuthData = L(a) || a || padding
AddAuthData = [0x00, 0x07, 0x8C, 0x1A, 0x0A, 0x02, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00]

PlaintextData = m || padding
PlaintextData = [0x20, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00]

AuthData = AddAuthData || PlaintextData
AuthData = [0x00, 0x07, 0x8C, 0x1A, 0x0A, 0x02, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x20, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00]

FlagsAuth = [Reserved = 0b0 ||Adata = 0b1 || (M-2)/2 = 0b001 || (L-1) = 0b001 → 0x49]

B0 = [Flags =0x49 || Nonce N = 0x11 0x22 0x33 0x44 0x55 0x66 0x77 0x88 0x02 0x00 0x00 0x00 0x05 || l(m) = 0x00 0x01]

B1 = [0x00, 0x07, 0x8C, 0x1A, 0x0A, 0x02, 0x00, 0x00, 0x00, 0x20, 0x00, 0x00, 0x00, 0x00, 0x00,

0x00]
B2 = [0x20, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00]

FlagsEncrypt = [Reserved = 0b0 || Reserved = 0b0 || 0b000 || (L-1) = 0b001 → 0x01]

Ai = [FlagsEncrypt = 0x01 || Nonce N = 0x11 0x22 0x33 0x44 0x55 0x66 0x77 0x88, 0x02, 0x00, 0x00, 0x00, 0x05 || Counter = 0x00 0x0i]

M1 = [0x20, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00]

Result
U = **0x1B36A2D2**
MIC = FULL U = 0x1B36A2D2 = [0xD2, 0xA2, 0x36, 0x1B]

Cipher = **0x7E**

A.1.5.10 Security test vectors for *ApplicationID* = 0b010 and an individual OOB key

A.1.5.10.1 [34]Common settings

- GP Security Key = [0xC0 , 0xC1 , 0xC2 , 0xC3 , 0xC4 , 0xC5 , 0xC6 , 0xC7 , 0xC8 , 0xC9 , 0xCa , 0xCb , 0xCc , 0xCd , 0xCe , 0xCf] = 0xCFCECDCCCBCAC9C8C7C6C5C4C3C2C1C0
- MAC fields:
  - Dest PANId = 0xffff
  - Dest Addr = 0xffff
  - MAC SeqNum = 0x02
- NWK fields:
  - NWK FC := [Ext NWK Header = 0b1 || Auto-Commissioning =0b0|| Zigbee Protocol 0b0011 || Frame type =0b00] → [0b10001100] 0x8c
  - GPD IEEE address = 0x8877665544332211
  - Endpoint = 0x0A
  - Security Frame Counter = 0x00000002
- Application fields:
  - GPD CommandID = 0x20 (OFF)
  - No data payload

A.1.5.10.2 [35]SecurityLevel=0b10

A.1.5.10.2.1 Transmitted packet

Transmitted packet = MAC FC || MAC header || GP stub NWK header || Payload || MIC
Transmitted packet
12 41 C8 02 FF FF FF FF 11 22 33 44 55 66 77 88 8C 32 0A 02 00 00 00 20 BD D2 CA AB

[34] CCB #2346; Resolution added in 15-02014-011
[35] CCB #2346; Resolution added in 15-02014-011

A.1.5.10.2.2 Inputs

Extended NWK FC = [Direction = 0b0 || RxAfterTx = 0b0 || SecurityKey = 0b1 || SecurityLevel = 0b10 || ApplID = 0b010] →0x32

SrcID field: absent;

A.1.5.10.2.3 Security Calculation

Nonce N = [0x11, 0x22, 0x33, 0x44, 0x55, 0x66, 0x77, 0x88, 0x02, 0x00, 0x00, 0x00, 0x05]

AddAuthData = L(a) || a || padding
AddAuthData = [0x00, 0x08, 0x8C, 0x32, 0x0A, 0x02, 0x00, 0x00, 0x00, 0x20, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00]

Flags = [Reserved = 0b0 ||Adata = 0b1 || (M-2)/2 = 0b001 || (L-1) = 0b001 → 0x49]

B0 = [Flags =0x49|| Nonce N = 0x11 0x22 0x33 0x44 0x55 0x66 0x77 0x88, 0x02, 0x00, 0x00, 0x00, 0x05 || 0x00 0x00]

[36]**Result**
U = **0xABCAD2BD**
MIC = FULL U = 0xABCAD2BD = [0xBD 0xD2 0xCA 0xAB]

A.1.5.10.3 [37]SecurityLevel=0b11

A.1.5.10.3.1 Transmitted packet

Transmitted packet = MAC FC || MAC header || GP stub NWK header || Payload || MIC

Transmitted packet

12 41 C8 02 FF FF FF FF 11 22 33 44 55 66 77 88 8C 3A 0A 02 00 00 00 7E DA 01 EE 3E

A.1.5.10.3.2 Inputs

Extended NWK FC = [Direction = 0b0 || RxAfterTx = 0b0 || SecurityKey = 0b1 || SecurityLevel = 0b11 || ApplID = 0b010] →0x3A

SrcID field: absent;

A.1.5.10.3.3 Security Calculation

Nonce N = [0x11, 0x22, 0x33, 0x44, 0x55, 0x66, 0x77, 0x88, 0x02, 0x00, 0x00, 0x00, 0x05]

AuthData = [0x00, 0x07, 0x8C, 0x3A, 0x0A, 0x02, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x20, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00]

FlagsAuth = [Reserved = 0b0 ||Adata = 0b1 || (M-2)/2 = 0b001 || (L-1) = 0b001 → 0x49]

B0 = [Flags =0x49 || Nonce N = 0x11 0x22 0x33 0x44 0x55 0x66 0x77 0x88 0x02 0x00 0x00 0x00 0x05 || l(m) = 0x00 0x01]

[36] CCB #2626; Resolution added in 16-02607-025
[37] CCB #2346; Resolution added in 15-02014-011

B1 = [0x00, 0x07, 0x8C, 0x3A, 0x0A, 0x02, 0x00, 0x00, 0x00, 0x20, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00]
B2 = [0x20, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00]

FlagsEncrypt = [Reserved = 0b0 || Reserved = 0b0 || 0b000 || (L-1) = 0b001 → 0x01]

Ai = [FlagsEncrypt = 0x01 || Nonce N = 0x11 0x22 0x33 0x44 0x55 0x66 0x77 0x88, 0x02, 0x00, 0x00, 0x00, 0x05 || Counter = 0x00 0x0i]

M1 = [0x20, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00]

Result
U = **0x3EEE01DA**
MIC = FULL U = 0x3EEE01DA = [0xDA, 0x01, 0xEE, 3E]

Cipher = **0x7E**

A.1.5.11 Security test vectors for *ApplicationID* = 0b010 and bidirectional operation

A.1.5.11.1 Common settings

For all frames

- NWK *Frame Type* sub-field = 0b00
- *Zigbee Protocol Version* sub-field = 0b0011
- *Auto-Commissioning* sub-field = 0b0
- *NWK Frame Control Extension* sub-field = 0b1
- GPD IEEE address = 0x8877665544332211
- Endpoint = 0x0A
- Security Frame Counter = 0x00000002
- Security Key = { 0xC0 0xC1 0xC2 0xC3 0xC4 0xC5 0xC6 0xC7 0xC8 0xC9 0xCA 0xCB 0xCC 0xCD 0xCE 0xCF }

For outgoing frames (from GPP/GPS to GPD)

- *RxAfterTx* sub-field = 0b0
- Direction sub-field = 0b1
- MAC Seq Nbr = 39
- GPD CommandID = 0xF1 (Write Attributes)
- GPD Command payload = 0x00 0x03 0x00 0x05 0x00 0x00 0x21 0x0a 0x00

A.1.5.11.2 Security test vectors for a shared key

For all test vectors with a shared security key:

- *SecurityKey* sub-field of *Extended NWK Frame Control* field = 0b0 (shared key)

A.1.5.11.2.1 SecurityLevel = 0b10

Outgoing frame (GPP/GPS to GPD)

01 0c 02 ff ff 11 22 33 44 55 66 77 88 8C 92 0A 02 00 00 00 F1 00 03 00 05 00 00 21 0A 00 03 48 0D 4D

A.1.5.11.2.2 SecurityLevel = 0b11

<u>Outgoing frame (GPP/GPS to GPD)</u>

01 0c 02 ff ff 11 22 33 44 55 66 77 88 8C 9A 0A 02 00 00 00 99 2C 16 34 58 B4 A6 EF 6D 12 89 2F 5E 1F

A.1.5.11.3 Security test vectors for an individual OOB key

For all test vectors with an individual key:

- *SecurityKey* sub-field of *Extended NWK Frame Control* field = 0b1 (individual key)

A.1.5.11.3.1 SecurityLevel = 0b10

<u>Outgoing frame (GPP/GPS to GPD)</u>

01 0c 02 ff ff 11 22 33 44 55 66 77 88 8C B2 0A 02 00 00 00 F1 00 03 00 05 00 00 21 0A 00 F1 3D 2A D9

A.1.5.11.3.2 SecurityLevel = 0b11

<u>Outgoing frame (GPP/GPS to GPD)</u>

0x01 0x0c 0x02 0xff 0xff 0x11 0x22 0x33 0x44 0x55 0x66 0x77 0x88 8C BA 0A 02 00 00 00 99 2C 16 34 58 B4 A6 EF 6D 12 3E 56 82 47

A.1.5.12 Security test vectors for key derivation

A.1.5.12.1 Derived individual GPD key

Input:

GPD IEEE address = 0x8877665544332211;

Endpoint = 0x0A; (not used for key derivation)

GPD Group Key = {0xc0, 0xc1, 0xc2, 0xc3, 0xc4, 0xc5, 0xc6, 0xc7, 0xc8, 0xc9, 0xca, 0xcb, 0xcc, 0xcd, 0xce, 0xcf};

Output:

Derived individual GPD key = {0x8a, 0xe7, 0x5b, 0x07, 0x5f, 0x7a, 0x13, 0x23, 0x06, 0x08, 0xff, 0x7e, 0x93, 0x07, 0x97, 0x6d};

A.1.5.13 Security test vectors for *ApplicationID* = 0b010 and TC-LK protection

A.1.5.13.1 OOB key in Commissioning GPDF for GPD IEEE address = 0x8877665544332211

<u>Input:</u>

GPD IEEE address = 0x8877665544332211

Endpoint = 0x0A; (not used for TC-LK protection)

OOB Key = {0xC0 0xC1 0xC2 0xC3 0xC4 0xC5 0xC6 0xC7 0xC8 0xC9 0xCA 0xCB 0xCC 0xCD 0xCE 0xCF}

TC-LK = {0x5A 0x69 0x67 0x42 0x65 0x65 0x41 0x6C 0x6C 0x69 0x61 0x6E 0x63 0x65 0x30 0x39}

Security frame counter – irrelevant;

<u>Processing:</u>

Nonce = {0x11 0x22 0x33 0x44 0x55 0x66 0x77 0x88 0x11 0x22 0x33 0x44 0x05 }

Header = {0x11 0x22 0x33 0x44}

Output:

TC-LK protected OOB key = {0x2D 0xF0 0x67 0xAF 0xCD 0x4D 0x8C 0xF0 0xF5 0x2E 0x6C 0x85 0x8F 0x31 0x4E 0x22}

GPDkeyMIC = {0x3F 0x9A 0xE0 0xB5}

A.1.5.13.2 Shared key in Commissioning Reply GPDF for GPD IEEE address = 0x8877665544332211

Input:

GPD IEEE address = 0x8877665544332211

Endpoint = 0x00; (not used for TC-LK protection)

Shared Key = {0xC0 0xC1 0xC2 0xC3 0xC4 0xC5 0xC6 0xC7 0xC8 0xC9 0xCA 0xCB 0xCC 0xCD 0xCE 0xCF}

TC-LK ={0x5A 0x69 0x67 0x42 0x65 0x65 0x41 0x6C 0x6C 0x69 0x61 0x6E 0x63 0x65 0x30 0x39}

Security frame counter from the GPDF that triggers Commissioning Reply *creation*, not *sending* = 2;

Processing:

Nonce = {0x11 0x22 0x33 0x44 0x55 0x66 0x77 0x88 0x03 0x00 0x00 0x00 0xC5}

Header = {0x11 0x22 0x33 0x44}

Output:

TC-LK protected shared key = { 0x2D 0x23 0x8F 0x58 0x07 0x1C 0x07 0x8A 0xB0 0x5C 0x23 0x5E 0x4D 0xED 0xDF 0x3B }

GPDkeyMIC = {0xDE 0xF5 0x18 0x7D}

Frame Counter = {0x03 0x00 0x00 0x00}

A.1.5.14 dLPED stub

Out of scope for the current document, to be specified by a separate LPED document.

A.1.6 GPD specification

The Green Power Device (GPD) is not required to implement any part of the Zigbee stack or the GP stub as described above. It implements the minimum MAC and stack functionality that allows it to support the required application functionality as defined per GPD device type in A.4.

Still, the following minimum implementation requirements need to be considered, to ensure interoperability with the GP infrastructure devices.

A.1.6.1 Frame format

As defined in A.1.4. Command payloads as defined in A.4.

A.1.6.2 GPD addressing

GPD is not part of the Zigbee network therefore it does not have the short (16-bit) address. The GPD SHALL support one of the unique identifications specified below; it SHALL NOT change the identification during its lifetime in a system.

A.1.6.2.1 ApplicationID = 0b000

If GPD supports *ApplicationID* = 0b000, the GPD is identified by the 4B SrcID. If it has enough energy, the GPD MAY in addition include its IEEE address in the MAC header of the GPDF.

The SrcID SHALL be globally unique. They are managed by the Zigbee Alliance, as described in [9]. The following SrcID values are reserved: 0x00000000 (used for none/undefined), 0xffffffff (used for all/any), and all in the range 0xfffffff9-0xfffffffe (reserved).

In the current Green Power specification, for the Green Power Devices there is no construct equivalent to Zigbee endpoints. However, it is possible for a GPD to use different SrcID values for each logical device existing on a GPD.

If a GPD has to support multiple identical device descriptions (e.g. an on/off switch with two rockers), each device description SHALL correspond to unique SrcID. If a GPD has to support multiple, but different device descriptions, it is left to the implementers of this specification to decide whether to use one or multiple SrcID. Please note, that proxies perform filtering and tunneling based solely on the SrcID.

A.1.6.2.2 ApplicationID = 0b010

If GPD supports *ApplicationID* = 0b010, the GPD is identified by its IEEE address. In addition, the *Endpoint* field is always present (see sec. A.1.4.1.5). The *Endpoint* field can be used to uniquely identify each of the multiple logical devices sharing the same GPD radio.

Implementers are free to choose the identifier for the *Endpoint(s)* from the non-reserved range (see sec. A.1.4.1.5).

A.1.6.3 GPD bidirectional operation

If the GPD is capable of bidirectional operation, it SHALL use the following constants.

If a GPD is addressable by GPD IEEE address (i.e. *ApplicationID* = 0b0101), then the GPD capable of bidirectional communication SHALL be capable of receiving GPDF addressed both to the unique endpoint numbers supported by this GPD, and to endpoint 0xff.

A.1.6.3.1 gpdRxOffset

The *gpdRxOffset* is the time, measured from the start of the transmission of the first frame in the GPFS with *RxAfterTx* sub-field set to 0b1, after which an Rx-capable GPD will enable its radio for reception.

It has fixed value of 20 milliseconds.

For explanation on GPFS usage, please see sec. A.1.7.2.1.

A.1.6.3.2 gpdMinRxWindow

The *gpdMinRxWindow* is minimal duration of the reception window of an Rx-capable GPD.

[38]GPD vendors SHALL implement reception window duration that is equal to at least the sum of the *gpMaxTxOffsetVariation*, the actual duration of the triggering GPFS[39], and the duration corresponding to the actual GPD frame size to be received by this GPD, if substantially longer than the triggering GPDF[40].

Note: the Rx-capable GPDs SHALL have energy budget that allows for processing the received frame, e.g. non-volatilely store the supplied parameters.

[38] CCB #2210; Resolution added in 15-02014-007; incl. errata ballot comment #1037, added in 15-02014-008;
[39] Errata ballot comment #1037
[40] Errata ballot comment #1037

A.1.6.3.3 [41]GPFS duration

The GPFS duration, measured from the start of transmission of the first frame in the sequence to the end of transmission of the last frame in the sequence, SHALL NOT exceed:

- 7ms for GPFS with RxAfterTx = 0b1;
- 5ms for GPFS with *RxAfterTx* = 0b0.

A.1.6.4 GPD security parameters

A.1.6.4.1 gpdSecurityLevel

The *gpdSecurityLevel* parameter indicates the security level used by this GPD. It can take the values as defined in Table 11.

The supported *gpdSecurityLevel* is dependent on the energy capabilities of a particular GPD. A GPD is assumed to support only one *gpdSecurityLevel.*

According to the current version of the specification, only GPD that support *gpdSecurityLevel* = 0b10 or higher AND support TC-LK protection (as indicated by the *GPDkeyEncryption* sub-field of the *Extended Options* field of the GPD Commissioning command) of the GPD key, if exchanged over the air, can be certified.

A.1.6.4.2 gpdSecurityKeyType

The type of security key with which the GPD was programmed. This parameter can take the values as defined in Table 53.

A.1.6.4.3 gpdSecurityKey

The security key itself.

Note: if the GPD device comes with an OOB individual key, then it MAY need to be stored in addition to the key used in the operational network.

A.1.6.4.4 gpdSecurityFrameCounter

The frame counter, used as part of the AES Nonce (see A.1.5.3.2).

The new frame counter value SHALL be stored immediately after usage, before the GPD starts transmitting the protected frame.

A GPD SHALL use one and the same frame counter for commissioning and operational mode, irrespective of the security levels used in both modes. Thus, when switching between the modes, the GPD continues with the next frame counter value.

The GPD SHALL preserve the security frame counter across “factory resets” (if implemented) and when being commissioned/decommissioned on different networks. [42]The only time the GPD SHALL reset the frame counter to zero is [43]if upon GPD Commissioning Reply command reception the security frame counter of the GPD is larger than 0x80000000 AND the type or value of the supplied key differs from the key currently used.

For *gpdSecurityLevel* 0b10 and 0b11, the *MAC sequence number* field SHOULD carry the 1LSB of the *gpdSecurityFrameCounter*.

A.1.6.4.5 GPD security processing for transmitted GPDF

See section A.1.5.3.2- A.1.5.3.4 and A.1.5.4.

[41] CCB #2210; Resolution added in 15-02014-007
[42] Note: Sink behavior to be specified as part of next GP release (v1.1).
[43] CCB #2419, resolved in 15-02014r010

A.1.6.4.6 GPD security processing for received GPDF

If the GPD is capable of bidirectional operation, the GPD SHALL perform the following checks on GPDF reception and drop the GPDF if any of those checks fails:

- The *ApplicationID* sub-field SHALL be set to the value supported by this GPD (0b000 or 0b010);
- The *Direction* sub-field SHALL be set to 0b1
- The value of the unique GPD ID in the received GPDF SHALL correspond to the GPD ID this device was programmed with.

Furthermore,

- if gpdSecurityLevel = 0b00, the GPD SHALL accept any *MAC sequence number* value;
- if gpdSecurityLevel = 0b10 – 0b11
  - The SecurityLevel, SecurityKeyType, and SecurityFrameCounter value in the received frame SHALL be exactly as for the triggering frame
  - The security processing SHALL be successful.

A.1.7 GPD implementation considerations

A.1.7.1 MAC frame control field

The Frame Control field of a GPDF MAC frame SHALL be formatted as illustrated in Figure 12. The bottom row of Figure 12 contains the recommended settings for minimum-functionality GPDs.

Bits: 0–2	**3**	**4**	**5**	**6**	**7–9**	**10–11**	**12–13**	**14–15**
Frame Type	Security Enabled	Frame Pending	Acknowl-edgment Request	Intra-PAN	Reserved	Destination Addressing Mode	Reserved	Source Addressing Mode
001	0	0	0/1	0	000	10	00	00

Figure 12 – GPDF MAC Frame Control Field Format

A.1.7.1.1 MAC sequence number field

GPDs that do not support security (*gpdSecurityLevel* = 0b00) may support random or incremental sequence numbers. That doesn’t make any functional difference in the system, since the receiving proxy/sink does NOT use it for security or freshness check, but only for duplicate filtering.

For GPDs that support security (*gpdSecurityLevel* >= 0b10), see sec. A.1.6.4.4.

A.1.7.1.2 MAC addressing fields

To remain IEEE 802.15.4 compliant, while minimizing the GPDF length, only the destination PANID and destination address fields MAY be present. Both SHALL be set to a value 0xffff, indicating unspecified/broadcast.

If the GPD has more energy available, it MAY include its IEEE address or the PANId of the Zigbee network.

Please note that usage of individual PANId MAY lead to device disconnection and need for re-commissioning in case of PANId change.

A.1.7.2 Energy budget of GPD

This specification covers a range of energy-restricted devices, from those with minimum energy budget (in the order of hundreds of μJ), with a typical example of electro-mechanical switch, up to devices with constant energy supply, with a typical example of a solar-powered sensor.

The GPD vendors are allowed to use the available energy budget in a way best fitting their application, choosing the required Green Power functionality (e.g. security, bidirectional commissioning, bidirectional communication, CSMA/CA usage, etc.).

According to the current version of the specification, only GPD that support *gpdSecurityLevel* = 0b10 or higher AND support TC-LK protection (as indicated by the *GPDkeyEncryption* sub-field of the *Extended Options* field of the GPD Commissioning command) of the GPD key, if exchanged over the air, can be certified.

A.1.7.2.1 [44]Energy budget and medium access

GPD devices with very restricted energy budget MAY skip CSMA/CA (incl. CCA) and repeat the Green Power Device Frame multiple times instead, to achieve the best possible reliability with the energy constraints given. Such a series of Green Power Device Frames, which are identical, incl. identical MAC sequence number, is then called Green Power Frame Sequence (GPFS). The number of frames in a GPFS and time spacing between them are left up to the implementer. The only limitation is the GPFS maximum duration as specified in section A.1.6.3.3.

The receiver only needs to act upon one of the frames in each GPFS; the others are dropped on reception as duplicates.

Devices with higher energy budget are recommended to perform CSMA/CA, so that they do not interfere with other communication on the same channel. This is especially recommended, if the device is to communicate frequently (e.g. a periodically reporting sensor).

A.1.7.3 GPD commissioning

GPD can send a Commissioning GPDF, to facilitate the commissioning process.

Otherwise, if the GPD is not capable of sending the Commissioning GPDF, the GPD SHALL be capable of sending at least one Data GPDF with the *Auto-Commissioning* flag set to 0b1, and the commissioning is performed with this/these Data GPDF. If the GPD is capable of being put in commissioning mode, it MAY set the *Auto-Commissioning* flag temporarily; otherwise the GPD SHALL permanently sets the *Auto-Commissioning* flag to 0b1 for this/these Data GPDF.

According to the current version of the specification, only GPD that support *gpdSecurityLevel* = 0b10 or higher AND support TC-LK protection (as indicated by the *GPDkeyEncryption* sub-field of the *Extended Options* field of the GPD Commissioning command) of the GPD key, if exchanged over the air, can be certified.
Since the GPD using the Auto-*Commissioning* = 0b1 do not exchange Commissioning (Reply) GPDF carrying the security key, such GPD would require out-of-band key establishment with the sink (out of scope for the current specification).

[44] CCB #2210; Resolution added in 15-02014-007; Incl. Errata ballot comment #1038, added in 15-02014-008;

GPD can set the *RxAfterTx* sub-field to 0b1 in the Commissioning GPDF, to facilitate bidirectional commissioning, especially to allow the network to deliver some configuration parameters (e.g. key, channel) to the GPD. The GPD SHOULD only set the *RxAfterTx* sub-field in the Commissioning GPDF, if it expects a response, i.e. if at least one of the sub-fields *PANId request* sub-field or *GPsecurityKeyRequest* is set to 0b1. The GPD SHOULD only request the key by setting *GPsecurityKeyRequest* to 0b1, if it supports security, i.e. if the *SecurityLevelCapabilities* sub-field of the *Extended Options* field of the GPD Commissioning command is set to 0b10 or 0b11.
A GPD setting *GPsecurityKeyRequest* to 0b1 SHALL also set the *GPDkeyPresent* sub-field of the *Extended Options* field of the Commissioning GPDF and include correctly protected *GPDkey* field. This is done to allow the Combo Basic devices according to the current specification, which may not be capable of delivering a shared key, to use the OOB key instead.

A GPD supporting bidirectional commissioning and *gpdSecurityLevel* 0b10 or 0b11 MAY choose to only provide the OOB key, i.e. set the *GPsecurityKeyRequest* sub-field of the *Options* field to 0b0.

A GPD supporting bidirectional commissioning is recommended to send the last frame of the bidirectional commissioning exchange, the Success GPDF, more than one time, to increase the probability of correct reception. If more than one Success GPFS is sent, and if *gpdSecurityLevel* is set to 0b10 or 0b11, the security frame counter SHALL be incremented for every transmission of a Success GPFS.

More on security usage during GPD commissioning can be found in A.3.9.2.

A.1.7.3.1 GPD bidirectional communication vs. Basic infrastructure

A GPD capable of bidirectional communication in operation may be instructed that the network only supports GP basic functionality, i.e. does not support bidirectional communication in operation. To accomplish this, the network sets to 0b1 the *Basic* sub-field of the *Channel* field of the GPD Channel Configuration command, following one of the GPD Channel Request command sent by the GPD.

A.1.7.3.2 [45]Commissioning vs. decommissioning/reset

GPD may need to be configured for use in a Zigbee network other than the one originally joined. Also, the GPD may need to be recommissioned, if the parameters of the Zigbee network the device operates in (esp. the operational channel or the shared key) change. There may also be a need to perform subsequent commissioning without prior reset, for example to pair the GPD with an additional sink.

GPDs which do not offer decommissioning/reset functionality SHALL start each commissioning exchange by toggling through the supported channels according to the supported commissioning procedure (using GPD Channel Request command, if bidirectional commissioning is supported, or using GPD Commissioning command with *RxAfterTx* = 0b0, if unidirectional commissioning is supported).

GPDs which do offer decommissioning/reset functionality MAY use the previously obtained commissioning knowledge (e.g. operational channel) until reset/decommissioned.

The GPD supporting subsequent commissioning, if capable of bidirectional commissioning, SHALL implement one of the following options for the subsequent commissioning: (i) repeating exactly the entire bidirectional commissioning procedure, but with the unprotected Commissioning GPDF carrying

[45] Generic switch commissioning guidelines, Zigbee document 16-02604-004
Dec 2016 SVE comment: https://workspace.zigbee.org/kws/groups/PRO_GP/comments/view_comment?comment_id=1012

an incremented *GPDoutgoingCounter* field and the encrypted security key of type and value as negotiated in the previous commissioning exchange OR (ii) performing a simplified unidirectional commissioning procedure, consisting of transmitting only on the operational channel of the network an unprotected Commissioning GPDF with *RxAfterTx* = 0b0 and the encrypted security key of type and value as negotiated in the previous commissioning exchange.
The GPD supporting subsequent commissioning, if only capable of unidirectional commissioning, SHALL implement one of the following options for the subsequent commissioning: (i) repeating exactly the entire unidirectional commissioning procedure, including channel toggling, but with the *GPDoutgoingCounter* field incremented and the encrypted security key of type and value as negotiated in the previous commissioning exchange OR (ii) performing a simplified unidirectional commissioning procedure, consisting of transmitting only on the operational channel of the network the Commissioning GPDF with *RxAfterTx* = 0b0, the *GPDoutgoingCounter* field incremented and the encrypted security key of type and value as negotiated in the previous commissioning exchange.
The security frame counter SHALL be incremented for each commissioning frame carrying it.
In case of generic switch functionality, the Commissioning GPDF MAY also carry another value of the *Current contact status* sub-field of the *Switch information* field.
GPD supporting a simplified procedure for the subsequent commissioning SHALL provide means for re-triggering the complete commissioning procedure, e.g. via prior decommissioning/reset.

After reset/decommissioning, the GPD SHALL be capable of performing the complete commissioning procedure, starting with toggling through the supported channels according to the supported commissioning procedure.

A.1.7.4 Configuration of network channel

During the commissioning procedure, the GPD is brought onto the operational channel of the Zigbee network.

If the GPD is capable of bidirectional commissioning, upon sending GPD Channel Request command the GPD will receive a GPD Channel Configuration command from the network. In addition to configuring the GPD with the operational channel of the network, the GPD Channel Configuration command also informs the GPD, using the *Basic* sub-field of the *Channel* field of the GPD Channel Configuration command, if the network (i.e. the sinks the GPD attempts to pair with and/or the forwarding proxy(s)) is capable of bidirectional communication in operation.

The GPD Channel Request SHALL be sent on more than one channel; the channel toggling can be done on each user commissioning action, or – if the GPD energy budget allows – automatically upon enabling the commissioning on the GPD. To shorten the channel finding process, the GPD MAY open one reception window only after transmitting multiple GPD Channel Request frames on different channels. All the GPD Channel Request transmissions belonging to the same reception window SHALL carry the same information in the *Channel Toggling Behavior* field. The *Auto-Commissioning* sub-field, in combination with the Maintenance *Frame Type* field used by the GPD Channel Request, indicates the GPDF position with respect to the reception window. The GPD Channel Request frame which will be followed by a reception window SHALL have *Auto-Commissioning* sub-field set to 0b0; it SHALL be sent on the *Rx Channel*, as indicated in the *Rx channel in the next attempt* sub-field of the *Channel Toggling Behavior* field of the GPD Channel Request belonging to the previous reception window. The GPD Channel Request frame which will be followed by further GPD Channel Request transmissions SHALL have the *Auto-Commissioning* sub-field set to 0b1.

When defining the channel toggling behavior for the GPD capable of bidirectional commissioning, and especially when selecting the receive channel(s), the vendors need to be aware that the appointed TempMaster spends up to 5 seconds on the *TransmitChannel* which is not the operational channel of the network. In particular constellations of receive channels (e.g. any channel, operational channel, any channel other than the operational channel), this may lead to the TempMaster proxy being absent from the operational channel at the time the GPD sends the first GPD Commissioning command, which can be problematic, if there is only one GP infrastructure device in GPD's range.
The vendors of the GPD capable of bidirectional commissioning can remedy this situation, e.g. by being able to re-send the GPD Commissioning command through/after the 5 seconds, by having a fixed receive channel; by waiting 5 seconds before changing the receive channel, etc.; if the vendors always choose a different receive channel, the probability of getting into this situation is rather low.

If the GPD is capable of bidirectional communication, it SHOULD be able to receive the GPD Channel Configuration command also during the operation. The GPD Channel Configuration command MAY be sent by the network in the event of network channel change.

The receiving GPD SHALL only execute such command, if it was appropriately secured (same security level and key as used by this GPD, fresh frame counter value).
This allows for avoiding GPD recommissioning.

A.1.7.5 Configuration of security key

During the commissioning procedure, the GPD and the network infrastructure agree on the security level and security use for subsequent communication protection.
If the GPD is Rx-capable, it MAY be able to receive the GPD Commissioning Reply command also during operation. The GPD Commissioning Reply command MAY be sent by the network in the event of change of the network-supplied security key.

The receiving GPD SHALL only execute such command, if it was appropriately secured (same security level and key as used by this GPD, fresh frame counter value).
This allows for avoiding GPD recommissioning.

The GPD SHALL only reset its security frame counter to 0x00000000 if upon GPD Commissioning Reply command reception the security frame counter of the GPD is larger than 0x80000000 AND the type or value of the supplied key differs from the key currently used. The GPD SHALL NOT reset the security frame counter upon transmission of GPD Decommissioning command. A GPD using an OOB key SHALL NOT reset the security frame counter at all.

If the GPD is capable of exchanging the security key encrypted, it SHALL set the *GPDkeyEncryption* sub-field of the *Extended Options* field of the GPD Commissioning command to 0b1, if at least one of the sub-fields *GPsecurityKeyRequest* or *GPDkeyPresent* of the GPD Commissioning GPDF command is set to 0b1. A GPD capable of exchanging the security key encrypted SHALL support receiving the key unprotected in the GPD Commissioning Reply command.

According to the current version of the specification, only GPD that support *gpdSecurityLevel* = 0b10 or higher AND support TC-LK protection (as indicated by the *GPDkeyEncryption* sub-field of the *Extended Options* field of the GPD Commissioning command) of the GPD key, if exchanged over the air, can be certified.

A.2 Zigbee core specification (r19) errata

This textual description of the GP compliance is provided for convenience of the reader.

The Green Power group would like to request for the following:

- Support of the GP feature to be **optional** for every Zigbee PRO device starting from the r20 release of the Zigbee core specification;
- Assignment of the (now reserved) Zigbee protocol version 0x3 for the Green Power Device Frame (GPDF);
- Assignment of a ClusterID for the Green Power cluster;
- Assignment of one of the reserved endpoint numbers (e.g. 242), to be used as fixed Green Power End Point. It does not need to be a dedicated endpoint; it can be shared with some other clusters.
- Assignment of profile-agnostic DeviceID values (analogous to the profile-agnostic Range extender, DeviceID = 0x0008) for the following GP infrastructure device types as defined in Table 13.

On behalf of the Low Power End Device group, the Green Power group would like to request:

- Inclusion of the NWKLPED-DATA.indication as a feature of the Zigbee core stack:
  - **Optional** for every Zigbee PRO device.

Furthermore, we would like to explicitly request Zigbee Routers to accept non-incremental NWK-level values in the *Sequence number* field of the Zigbee Network header for the consecutive packets with the same value of the *Source address* field of the Zigbee Network header (note: this request concerns the NWK header *Sequence number* field, and NOT the security *Frame Counter* field of the Auxiliary NWK Frame Header).

A.2.1 Notation

Black text – original specification text
~~Red text crossed over~~ - original text from the Zigbee r19 specification proposed to be removed
Red text – new proposed text
Headers - explanation for the r19 editors

A.2.2 All the changes are made against:

[22] Zigbee r19 specification: 1_053474r19_CSG-Zigbee-Specification.pdf, October 12, 2010.

A.2.3 GP Zigbee protocol version

A.2.3.1 Modify "Zigbee Protocol Version" definition in section 1.4.1.1 Conformance Levels, p. 7 of [22]

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 the Zigbee frames or a value of 0x03 for the Green Power frames. The protocol version support required by various Zigbee specification revisions appears below in Table 1.1.

A.2.3.2 Add a row to Table 1.1 Zigbee Protocol Versions, p. 7, of [22], above the 0x02 row

Specification	Protocol	Version Comment
Current	0x03	Green Power feature

A.2.3.3 Change the description below Table 1.1, p. 7, of [22]

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 or 0x02 or 0x03, and 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 GP stub (see Annex F), if it supports the Green Power, otherwise it SHALL drop them.

A.2.4 Support for Green Power EndPoint

A.2.4.1 Modify the "Device application" definition in section 1.4.1.2, p. 9, of [22]

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 Zigbee Alliance. The GreenPower cluster, if implemented, SHALL use endpoint 242.

A.2.4.2 Modify the “End application” definition in section 1.4.1.2, p. 10, of [22]

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

A.2.4.3 Modify section 2.1.2 “Application Framework”, p.18, of [22]

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 address from 1 to 254. Two additional endpoints are defined for APSDESAP usage: endpoint 0 is reserved for the data interface to the ZDO, and endpoint 255 is reserved for the data interface function to broadcast data to all application objects. Endpoints 241-254 are assigned by the Zigbee Alliance and SHALL NOT be used without approval. The GreenPower cluster, if implemented, SHALL use endpoint 242.
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 Zigbee Alliance. The GreenPower cluster, if implemented, SHALL use endpoint 242.

A.2.5 Support for proxy alias

A.2.5.1 Modify section 3.6.2.2 “Reception and Rejection”, p. 384, of [22]

3.6.2.2 Reception and Rejection

(…)
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.
The following data frames SHALL be passed to the next higher layer using the NLDE-DATA.indication primitive:
(…)

A.2.5.2 Modify section 3.6.2.1 “Transmission”, p. 383, of [22]

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 INVALID_REQUEST.
All frames handled by or generated within the NWK layer SHALL be constructed according to the general frame format specified in Figure 3.5 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 *nwkMaxDepth* attribute of the NIB (see clause 3.5).

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 ~~constructs~~ a new sequence number value for a NWK frame~~, either as a result of a request from the next higher layer to transmit a new NWK data frame or when it needs to construct a new~~
~~NWK layer command frame~~. ~~After being incremented, t~~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, (…)

A.2.5.3 Modify section 2.2.4.1.1 APSDE-DATA.request, p. 23, of [22]

A.2.5.3.1 Modify section 2.2.4.1.1.1 Semantics of the Service Primitive, p.23, of [22]

The semantics of this primitive are as follows:

```
APSDE-DATA.request {
DstAddrMode,
DstAddress,
DstEndpoint,
ProfileId,
ClusterId,
SrcEndpoint,
ADSULength,
ADSU,
TxOptions,
UseAlias,
AliasSrcAddr,
AliasSeqNumber,
```

RadiusCounter
}
Support of the additional parameters – UseAlias, AliasSrcAddr, AliasSeqNumb - in the APSDE-DATA.request primitive is required if GP feature is to be supported by the implementation.

A.2.5.3.2 Add to Table 2.2 APSDE-DATA.request Parameters, p.24, after the TxOptions parameter, the parameters UseAlias, AliasSrcAddr, AliasSeqNumb, defined as follows

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 *APS counter* value and NWK *Sequence number* value to be used for this APDU and NSDU. If the *UseAlias* parameter has a value of FALSE, the *AliasSeqNumb* parameter is ignored.

A.2.5.3.3 Modify section 2.2.4.1.1.3 Effect on Receipt, p. 25ff, of [22], as follows

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 must first be mapped to a corresponding 16-bit NWK address by using the *nwkAddressMap* attribute of the NIB (see Table 3.43). 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.
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.
If the DstAddrMode parameter has a value of 0x01, indicating group addressing, or the DstAddrMode parameter has a value of 0x00 and the corresponding binding table entry contains a group address, then the APSME will check the value of the *nwkUseMulticast* attribute of the NIB (see Table 3.44). If this attribute has a value of FALSE, then the delivery mode sub-field of the frame control field of the resulting APDU will be set to 0b11, the 16-bit address of the destination group will be placed in the group address field of the APS header of the outgoing frame, and the NSDU frame will be transmitted as a broadcast. A value of 0xfffd, that is, the broadcast to all devices for which macRxOnWhenIdle = TRUE, will be supplied for the DstAddr parameter of the NLDE-DATA.request that is used to transmit the frame. If the *nwkUseMulticast* attribute has a value of TRUE, then the outgoing frame will be transmitted using NWK layer multicast, with the delivery mode sub-field of the frame control field of the APDU set to 0b10, the destination endpoint field set to 0xff, and the group address not placed in the APS header.

The parameters UseAlias, AliasSrcAddr and AliasSeqNumb SHALL be used in the invocation of the NLDE-DATA.request primitive.
In addition, if the UseAlias parameter is set to TRUE, the AliasSeqNumb SHALL be copied into the APS counter field of the APS header. If the UseAlias parameter has a value of FALSE, then APS counter field of the APS header SHALL take the value as maintained by the APS.
If the UseAlias parameter has the value of TRUE, and the Acknowledged transmission field of the TxOptions parameter is set to 0b1, [46]then the Acknowledged transmission field of the TxOptions

[46] CCB #2158; Resolution added in 15-02014-005;

parameter SHALL be ignored.

If the TxOptions parameter specifies that secured transmission is required, the APS sub-layer SHALL use the security service provider (see sub-clause 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 subclause 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.
If NWK layer multicast is being used, the NonmemberRadius parameter of the NLDE-DATA.request primitive SHALL be set to *apsNonmemberRadius*.
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 larger than will fit in a single frame and fragmentation is not possible, then the ASDU is not transmitted and the APSDE SHALL issue the APSDE-DATA.confirm primitive with a status of ASDU_TOO_LONG. Fragmentation is not possible if either an acknowledged transmission is not requested, or if the fragmentation permitted flag in the TxOptions field is set to 0, or if the ASDU is too large to be handled by the APSDE.
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 sub-clause 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 documentation and/or 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.

A.2.5.4 Modify section 3.2.1.1 NLDE-DATA.request, p. 263ff, of [22]

A.2.5.4.1 Modify section 3.2.1.1.1, p. 264, of [22]

3.2.1.1.1 Semantics of the Service Primitive
The semantics of this primitive are as follows:
Table 3.2 specifies the parameters for the NLDE-DATA.request primitive.

```
NLDE-DATA.request {
DstAddrMode,
DstAddr,
NsduLength,
Nsdu,
NsduHandle,
UseAlias,
AliasSrcAddr,
AliasSeqNumber,
Radius,
NonmemberRadius,
DiscoverRoute,
SecurityEnable
}
```

Support of the additional parameters – UseAlias, AliasSrcAddr, AliasSeqNumb - in the APSDE-DATA.request primitive is required if GP feature is to be supported by the implementation.

A.2.5.4.2 Add to Table 3.2., p. 264ff, after the Radius parameter, the parameters UseAlias, AliasSrcAddr, AliasSeqNumb, defined as follows

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 denotesthat 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.

A.2.5.4.3 Modify 3.2.1.1.3, p. 265ff, of [22]

3.2.1.1.3 Effect on Receipt

If this primitive is received on a device that is not currently associated, the NWK layer will issue an NLDE-DATA.confirm primitive with a status of INVALID_REQUEST.

On receipt of this primitive, the NLDE first constructs an NPDU in order to transmit the supplied NSDU. If, during processing, the NLDE issues the NLDE-DATA. confirm primitive prior to transmission of the NSDU, all further processing is aborted. In constructing the new NPDU, the destination address field of the NWK header will be set to the value provided in the DstAddr parameter~~, and~~.

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 of the NWK header will have the value of the *macShortAddress* attribute in the MAC PIB.

The discover route sub-field of the frame control field of the NWK header will be set to the value provided in the DiscoverRoute parameter. If the supplied Radius parameter does not have a value of zero, then the radius field of the NWK header will be set to the value of the Radius parameter. If the Radius parameter has a value of zero, then the radius field of the NWK header will be set to twice the

value of the *nwkMaxDepth* attribute of the NIB.
If the UseAlias parameter has a value of TRUE, the sequence number field of the NWK header of the frame will be set to the value provided in the AliasSeqNumb parameter. If the UseAlias parameter has a value of FALSE, then ~~T~~the NWK layer will generate a sequence number for the frame as described in sub-clause 3.6.2.1. and the sequence number field of the NWK header of the frame will be set to this sequence number value.
The multicast flag field of the NWK header will be set according to the value of the DstAddrMode parameter. If the DstAddrMode parameter has a value of 0x01, the NWK header will contain a multicast control field whose fields will be set as follows:
• The multicast mode field will be set to 0x01 if this node is a member of the group specified in the DstAddr parameter.
• Otherwise, the multicast mode field will be set to 0x00.
• The non-member radius and the max non-member radius fields will be set to the value of the NonmemberRadius parameter.
Once the NPDU is constructed, the NSDU is routed using the procedure described in sub-clause 3.6.3.3 if it is a unicast, sub-clause 3.6.5 if it is a broadcast, or subclause 3.6.6.2 if it is a multicast. When the routing procedure specifies that the NSDU is to be transmitted, this is accomplished by issuing the MCPSDATA. request primitive with both the SrcAddrMode and DstAddrMode parameters set to 0x02, indicating the use of 16-bit network addresses. The SrcPANId and DstPANId parameters SHOULD be set to the current value of *macPANId* from the MAC PIB. The SrcAddr parameter will be set to the value of *macShortAddr* from the MAC PIB. The value of the DstAddr parameter is the next hop address determined by the routing procedure. If the message is a unicast, bit b0 of the TxOptions parameter SHOULD be set to 1 denoting that an acknowledgement is required. On receipt of the MCPS-DATA.confirm primitive on a unicast, the NLDE issues the NLDE-DATA.confirm primitive with a status equal to that received from the MAC sub-layer. Upon transmission of a MCPS-DATA. confirm primitive, in the case of a broadcast or multicast, the NLDE immediately issues the NLDE-DATA.confirm primitive with a status of success.12
If the *nwkSecurityLevel* NIB attribute has a non-zero value and the SecurityEnable parameter has a value of TRUE, then NWK layer security processing will be applied to the frame before transmission as described in clause 4.3. Otherwise, no security processing will be performed at the NWK layer for this frame. The security processing SHALL always be performed using device's own extended 64-bit IEEE address and OutgoingFrame 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.

A.2.6 Device_annce

A.2.6.1 Modify section 2.4.3.1.11.2, p. 111, of [22]

2.4.3.1.11.2 Effect on Receipt

(…)

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.

A.2.6.2 Modify section 2.4.4.1, p. 151, of [22]

2.4.4.1 Device and Service Discovery Server

Table 2.89 lists the commands supported by the Device and Service Discovery Server Services device profile. Each of these commands will be discussed in the following sub-clauses. 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 2.89 Device and Service Discovery Server Service Primitives

(…)

A.2.6.3 Modify section 3.6.1.9.2, p. 375, of [22]

3.6.1.9.2 Detecting Address Conflicts

After joining a network or changing address due to a conflict, a device SHALL send either a device_annc or initiate a route discovery prior to sending messages.

Upon receipt of a frame containing a 64-bit IEEE address in the NWK header, the contents of the *nwkAddressMap* attribute of the NIB and neighbor table SHOULD be checked for consistency.

If the destination address field of the NWK Header of the incoming frame is equal to the *nwkNetworkAddress* attribute of the NIB then the NWK layer SHALL check the destination IEEE address field, if present, even if it is the 0xff..ff 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~~16~~-bit address, the device has detected a conflict elsewhere in the network.

A.3 Green Power cluster

A.3.1 Overview

The Green Power cluster defines the format of the commands exchanged when handling GPDs.

A.3.2 GP infrastructure devices

GP infrastructure devices are the devices receiving the communication of the Green Power device (GPD). The Green Power specification defines two general types of the GP infrastructure devices: a sink which executes the GPD commands and a proxy which forwards the received GPD frames to the sinks.

The Device IDs used by GP specification and based on the general types mentioned above are defined in [10] and listed in Table 13; more detailed definitions of each DeviceID are provided in the remainder of this section.

According to the current specification, only Basic Proxy, Basic Combo and GP Commissioning Tool can be implemented; the other device types cannot be certified.

The implementation of GP Proxy Basic functionality is mandatory for Zigbee Routers seeking Zigbee 3.0 compliance.

While it is optional to implement the sink functionality for devices seeking Zigbee 3.0 compliance, vendors are strongly recommended by the Strategic Committee of the Zigbee Alliance to consider the use cases for GPD-controlled devices and to implement the sink functionality.

The Green Power cluster SHALL use ClusterID 0x0021.

The Green Power cluster SHALL be implemented on the reserved Green Power End Point - endpoint 0xF2 (242).

The reserved Green Power End Point SHALL use ProfileID 0xA1E0 in the Simple Descriptor, as well as in all Green Power cluster messages. The GP infrastructure devices SHALL NOT respond to communication using other ProfileIDs, including the common ProfileID = 0x0104 (see ProfileID matching rules of the Core specification).

In the Simple Descriptor, the GP infrastructure devices according to the current version of the GP specification SHALL set the Application device version field to 0x0.

Table 13 – List of GP infrastructure devices

	Device	Device ID
GP Generic	GP Proxy	0x0060
	GP Proxy Basic	0x0061
	GP Target Plus	0x0062
	GP Target	0x0063
	GP Commissioning Tool	0x0064
	GP Combo	0x0065
	GP Combo Basic	0x0066

A.3.2.1 GP Target device

According to the current specification, only Green Power Basic Proxy, Green Power Basic Combo and Green Power Commissioning Tool can be implemented.

In the current version of the specification, a GP Target can only be implemented on a ZED, because implementation of Basic Proxy is mandatory for Zigbee 3.0 ZR.

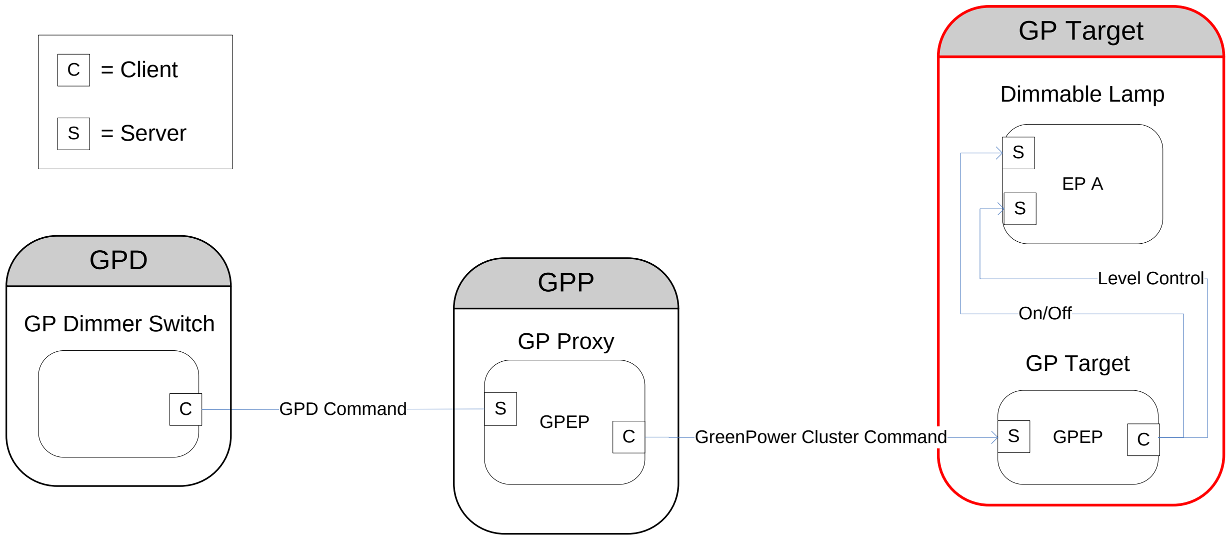
The functionality supported by the GP Target device is defined in Table 14.

Table 14 – Functionality of GP Target device

Server side (if supported by device)	**Client side**
Mandatory	
Selected Green Power cluster commands (see Table 23) and GP functionality (see Table 22)	Selected Green Power cluster commands (see Table 23) and GP functionality (see Table 22)
Optional	

The GP Target DeviceID (see Table 13) implements the server side of the Green Power cluster on the reserved end point, the Green Power EndPoint (see sec. A.3.6.1) with the selected commands of the client side of the Green Power cluster (see Table 23), as well as the selected server-side attributes (see sec. A.3.3.2), and has the following capabilities:

- Ability to receive any GP frame in tunneled mode;
- Ability to process or drop any incoming GP frame, received in tunneled mode, depending on pairings created during commissioning (i.e. ability to translate the relevant GP commands in the correct Zigbee ZCL format for its own applications);
- Ability to filter duplicate GP frames, received in tunneled mode;
- Optionally, depending on the desired communication mode, ability to acknowledge the GP frames received in the tunneled mode;
- Ability to create or delete at commissioning time the pairings between specific GPD and sink's own applications;
- Ability to (de-)register at the proxies (using GP Pairing command) at commissioning time in order to receive/stop receiving tunneled GP frames from desired GPD;
- Optionally, depending on the requirements of the supported applications, ability to configure selected parameters of the GPD during commissioning in tunneled mode.
- Optionally, depending on the requirements of the supported applications, ability to send messages back to the GPD during operation in tunneled mode.
- Optionally, depending on the requirements of the supported application, ability to use secured GPD communication.
- Optionally, depending on the requirements of the supported applications, ability to remove the GPD from the network (using GP Pairing command).



2523

2524 **Figure 13 – Example of GP Sink Basic device usage**

2525

A.3.2.2 GP Target+ device

According to the current specification, only Green Power Basic Proxy, Green Power Basic Combo and Green Power Commissioning Tool can be implemented.
In the current version of the specification, a GP Target can only be implemented on a ZED, because implementation of Basic Proxy is mandatory for Zigbee 3.0 ZR.

The functionality supported by the GP Target+ device is defined in Table 15.

Table 15 – Functionality of GP Target+ device

Server side	Client side
Mandatory	
Selected Green Power cluster commands (see Table 23) and GP functionality (see Table 22)	Selected Green Power cluster commands (see Table 23) and GP functionality (see Table 22)
	Rx GP stub
Optional	
Tx GP stub	

A GP Target+ DeviceID (see Table 13) requires implementation of both the server side of the Green Power cluster on the reserved end point, the Green Power EndPoint (see sec. A.3.6.1) with the selected commands of the client side of the Green Power cluster (see Table 23), the selected server-side attributes (see sec. A.3.3.2), as well as the GP stub. A GP Target+ device has all the capabilities of the GP Target device plus the ability of receiving GPD frames in the direct mode, which then requires:

- Ability to receive any GP frame both in direct mode and in tunneled mode (i.e. at both client and server side of the Green Power cluster);
- Ability to process or drop any incoming GP frame, received either in direct mode or in tunneled mode, depending on pairings created during commissioning;
- Ability to filter duplicate GP frames, received in both direct mode or in tunneled mode.
- Optionally, when bidirectional pairing or operation is to be supported, ability to send GPDF to the GPD in direct mode.

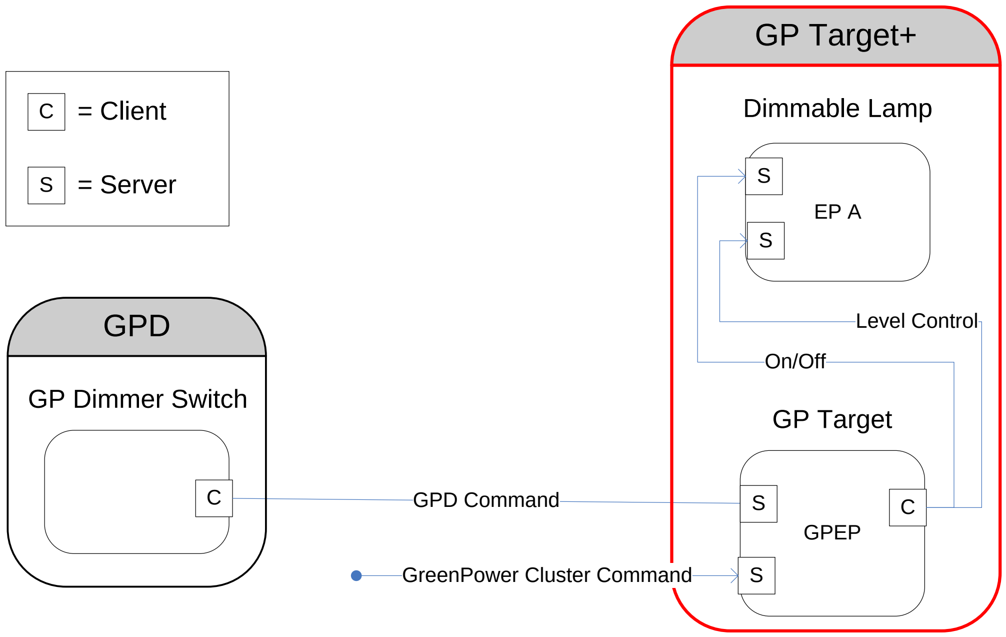


Figure 14 – Example of GP Target+ device usage

A.3.2.3 GP Proxy device

According to the current specification, only Green Power Basic Proxy, Green Power Basic Combo and Green Power Commissioning Tool can be implemented.

The functionality supported by the GP Proxy device is defined in Table 16.

Table 16 – Functionality of GP Proxy device

Server side	**Client side**
Mandatory	
Selected Green Power cluster commands (see Table 23) and GP functionality (see Table 21)	Selected Green Power cluster commands (see Table 23) and GP functionality (see Table 21)
Tx GP stub	Rx GP stub
Optional	

A Green Power Proxy is a normal Zigbee device, in most cases a ZR, which implements on its reserved end point, the Green Power EndPoint (see sec. A.3.6.1) the GP Proxy DeviceID (see Table 13) with the selected commands of the Green Power cluster (see Table 23), client-side attributes (see sec. A.3.4.2), and a GP stub. Green Power Proxy has the following GP proxy capabilities:

- Ability to receive any GP frame in direct mode when the proxy is in the radio range of the GPD;
- Ability to filter out duplicate GPDF received in direct mode (belonging to one GPFS);
- Ability to send to the registered sink devices a GP Notification command with the received GP frame;
- Ability to receive acknowledgements from the check if the sink has correctly received the tunneled GP frame if this communication mode is required at commissioning time;
- Ability to maintain a Proxy Table at commissioning time to register sink devices which are asking for GP frame forwarding service;
- Ability to update the Proxy Table based on the observed GP traffic in order to enable GP device mobility in the network;
- Ability to drop scheduled tunneling of GP frame, based on received GP commands related to the same GP frame.

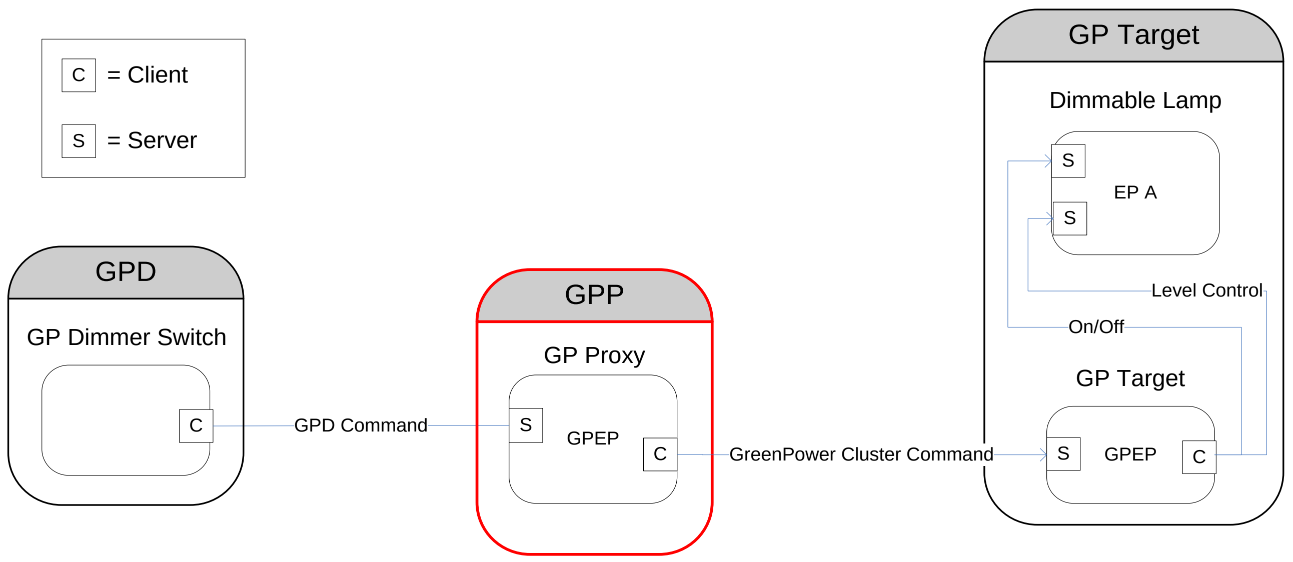


Figure 15 – Example of GP Proxy device usage

A.3.2.4 GP Combo device

According to the current specification, only Green Power Basic Proxy, Green Power Basic Combo and Green Power Commissioning Tool can be implemented.

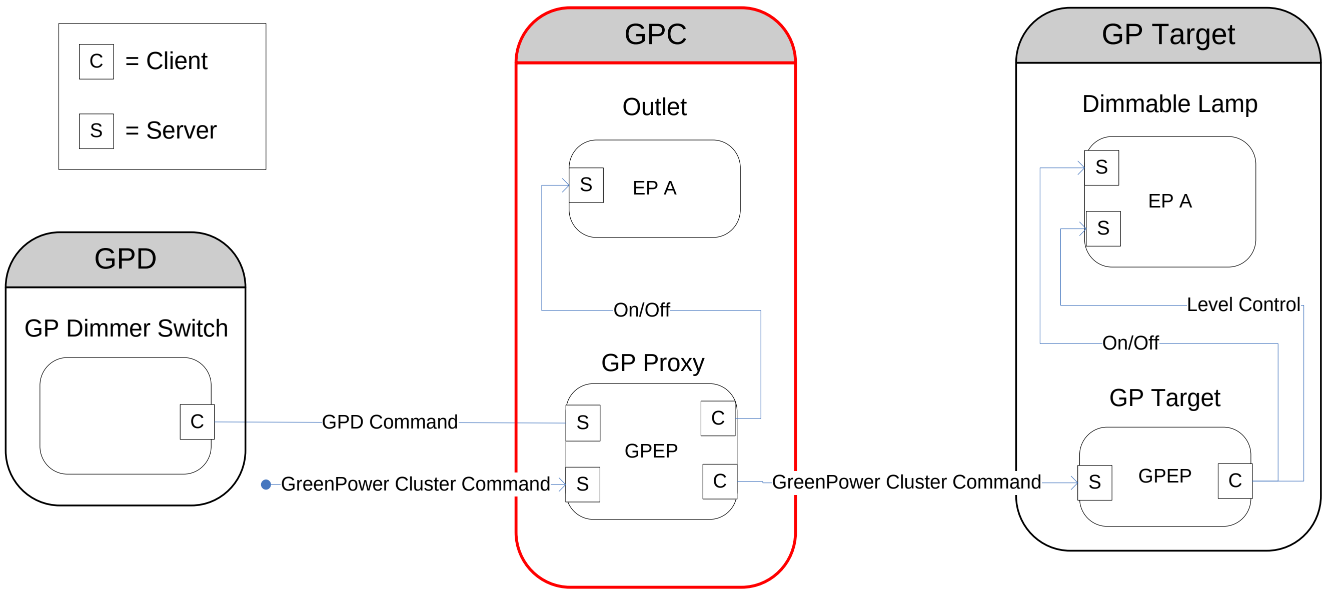
The functionality supported by the GP Combo device is defined in Table 17.

Table 17 – Functionality of GP Combo device

Server side	**Client side**
Mandatory	
Selected Green Power cluster (see Table 23) and GP functionality (see Table 21, Table 22)	Selected Green Power cluster commands (see Table 23) and GP functionality (see Table 21, Table 22)
Tx GP stub	Rx GP stub
Optional	

A Green Power Proxy can also be at the same time a sink device. In this case the device implements the GP Combo DeviceID (see Table 13) on the Green Power EndPoint (see sec. A.3.6.1) with selected server-side and client-side commands of the Green Power cluster (see Table 23), as well as the selected server-side attributes (see sec. A.3.3.2) and client-side attributes (see sec. A.3.4.2) and the GP stub. It has all the capabilities of both GPT+ and a Green Power Proxy, including the following:

- Ability to receive any GP frame both in direct mode and in tunneled mode (i.e. at both client and server side of the Green Power cluster);
- Ability to process or drop any incoming GP frame, received either in direct mode or in tunneled mode, depending on pairings created during commissioning;
- Ability to filter duplicate GP frames, received in both direct mode or in tunneled mode.

2588


2589 **Figure 16 – Example of GP Combo device usage**

2590

A.3.2.5 GP Commissioning Tool

According to the current specification, only Green Power Basic Proxy, Green Power Basic Combo and Green Power Commissioning Tool can be implemented. The functionality supported by the GP Commissioning Tool device is defined in Table 18.

Table 18 – Functionality of GP Commissioning Tool device

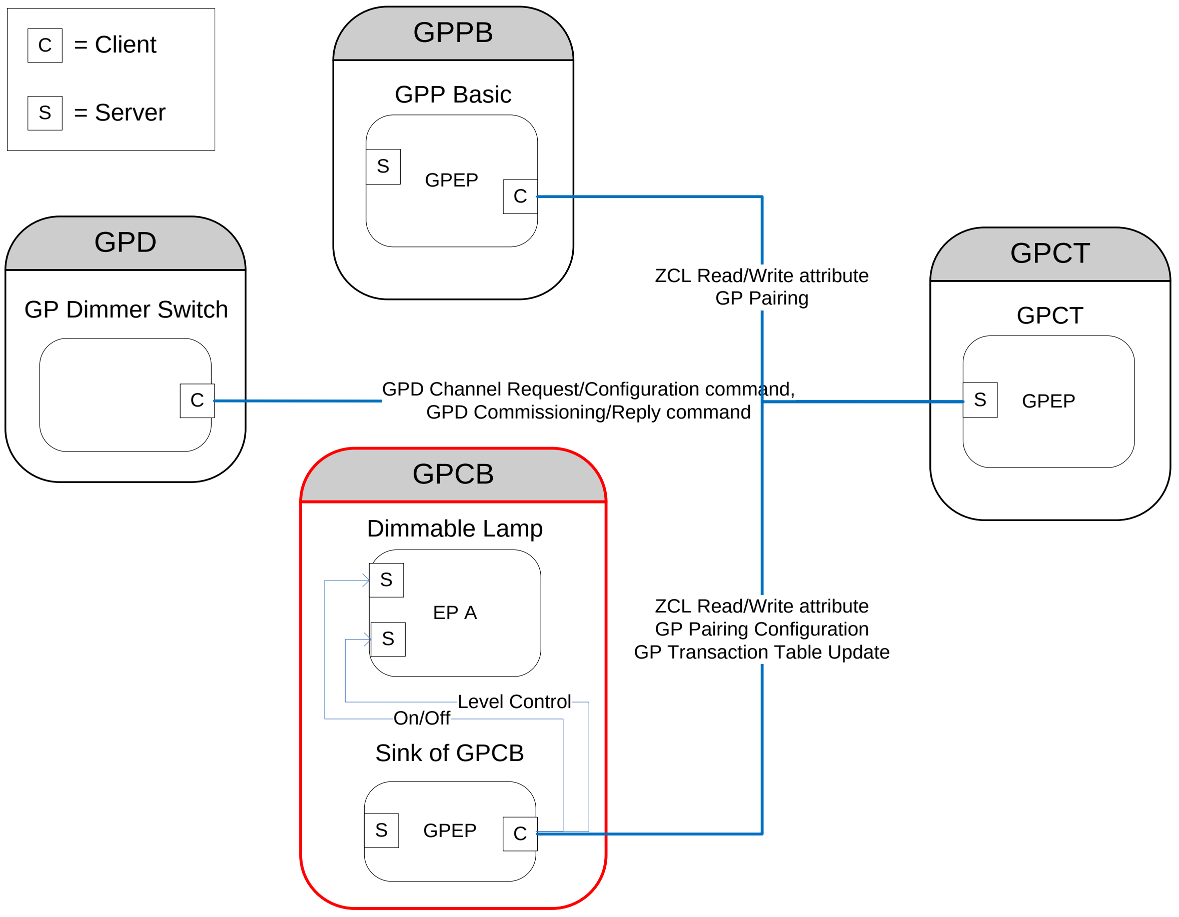
Server side	**Client side**
Mandatory	
Selected Green Power cluster commands	
Tx GP stub, Rx GP stub	
Optional	

A GPCT is a regular Zigbee device, in most cases a ZR, which implements [47]the GP Commissioning Tool DeviceID (see Table 13) on its reserved end point, the Green Power EndPoint (see sec. A.3.6.1) or another active endpoint that uses the Green Power ProfileID (0xA1E0).

[48]GPCT MAY have any of the following GP capabilities:

- Ability to receive any GPDF in direct mode when in the radio range of the GPD;
- Ability to transmit GPDF in direct mode when in the radio range of the GPD;
- Ability to process and generate GPD configuration commands (GPD Channel Request/Configuration, GPD Commissioning (Reply));
- Ability read/write Green Power cluster client/server attribute;
- Ability to send and receive GP configuration commands (GP Pairing, GP Pairing Configuration, GP Proxy Commissioning Mode, GP Translation Table Update, GP Translation Table Request, GP Translation Table Response);
- Ability to perform GPD application functionality matching.

[47] CCB #2372; Resolution added in 15-02014-011
[48] CCB #2372; Resolution added in 15-02014-011



2610

2611 **Figure 17 – Example of GP Commissioning Tool device usage**

2612

A.3.2.6 GP Proxy Basic device

According to the current specification, only Green Power Basic Proxy, Green Power Basic Combo and Green Power Commissioning Tool can be implemented.

The functionality supported by the GP Proxy Basic device is defined in Table 19.

Table 19 – Functionality of GP Proxy Basic device

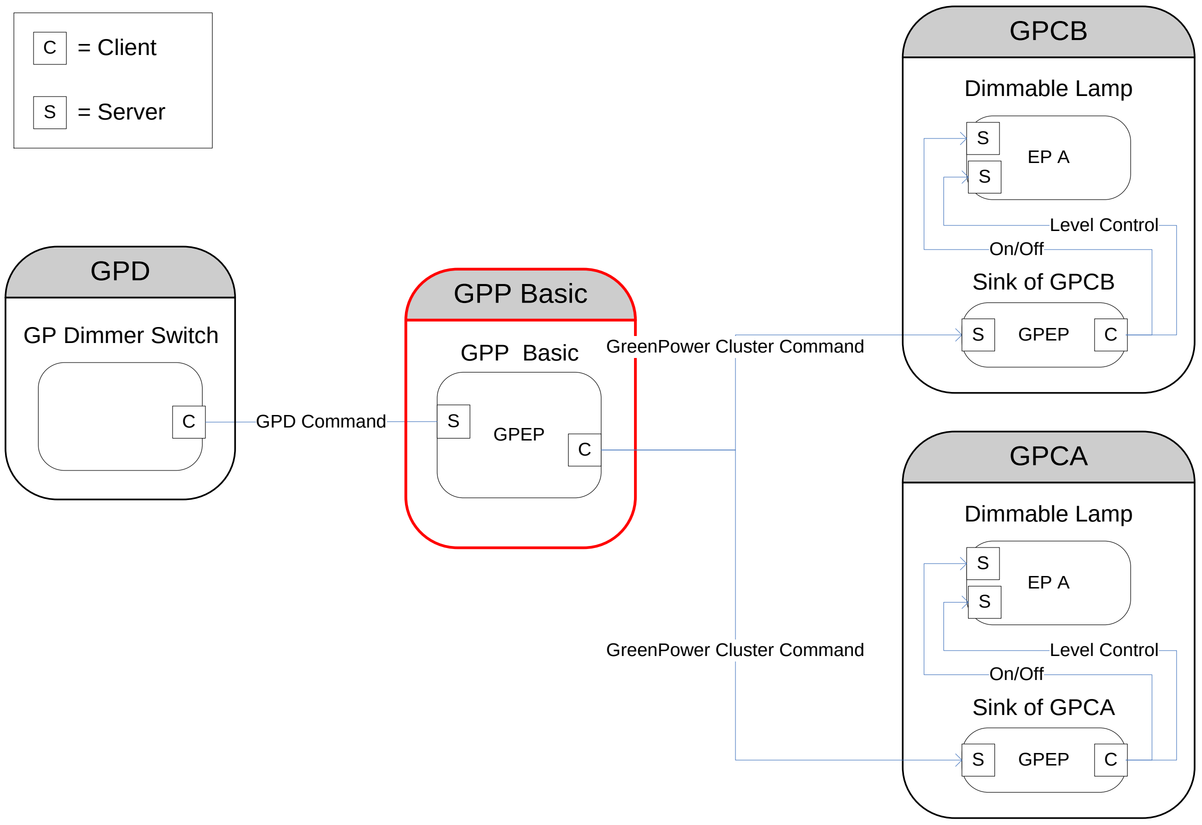
Server side	Client side
Mandatory	
	Selected Green Power cluster commands (see Table 23) and GP functionality (see Table 21)
	[49]Rx GP stub, Tx GP stub
Optional	

A Green Power Basic Proxy is a regular Zigbee device, in most cases a ZR, which implements on its reserved end point, the Green Power EndPoint (see sec. A.3.6.1) the Basic Proxy DeviceID (see Table 13) with the selected commands of the client side of the Green Power cluster (see Table 23), selected client-side attributes (see sec. A.3.4.2), and the reception functionality of the GP stub.

Basic Proxy has the following GP proxy capabilities (see also sec. A.3.2.8):

- Ability to receive any GP frame in direct mode when the Basic Proxy is in the radio range of the GPD;
- [50]Ability to transmit unprotected commissioning GPDF in direct mode when the Basic Proxy is in the radio range of the GPD;
- Ability to filter out duplicate GPDF received in direct mode (belonging to one GPFS);
- Ability to filter GPDFs by GPD ID of commissioned GPDs;
- Ability to security-process the GPDF before forwarding;
- Ability to send to the registered sink devices a groupcast GP Notification command with the received GPD command;
- Ability to maintain a Proxy Table to register GPD Ds of GPD and group addresses to enable GP frame forwarding.

[49] CCB #2114; Resolution added in 15-02014-002
[50] CCB #2114; Resolution added in 15-02014-002



2637
2638 **Figure 18 – Example of GP Proxy Basic device usage**

2639

A.3.2.7 GP Combo Basic device

According to the current specification, only Green Power Basic Proxy, Green Power Basic Combo and Green Power Commissioning Tool can be implemented.

The functionality supported by the GP Combo Basic device is defined in Table 20.

Table 20 – Functionality of GP Combo Basic device

Server side	Client side
Mandatory	
Selected Green Power cluster commands (see Table 23) and GP functionality (see Table 21, Table 22)	Selected Green Power cluster commands (see Table 23) and GP functionality (see Table 21, Table 22)
	[51]Rx GP stub, Tx GP stub
Optional	
Tx GP stub	

A Basic Combo implements the basic set of the combo functionality, i.e. basic set of proxy functionality, as depicted in Table 21and basic set of sink functionality, as depicted in Table 22, as well as the selected server-side attributes (see sec. A.3.3.2) and client-side attributes (see sec. A.3.4.2).

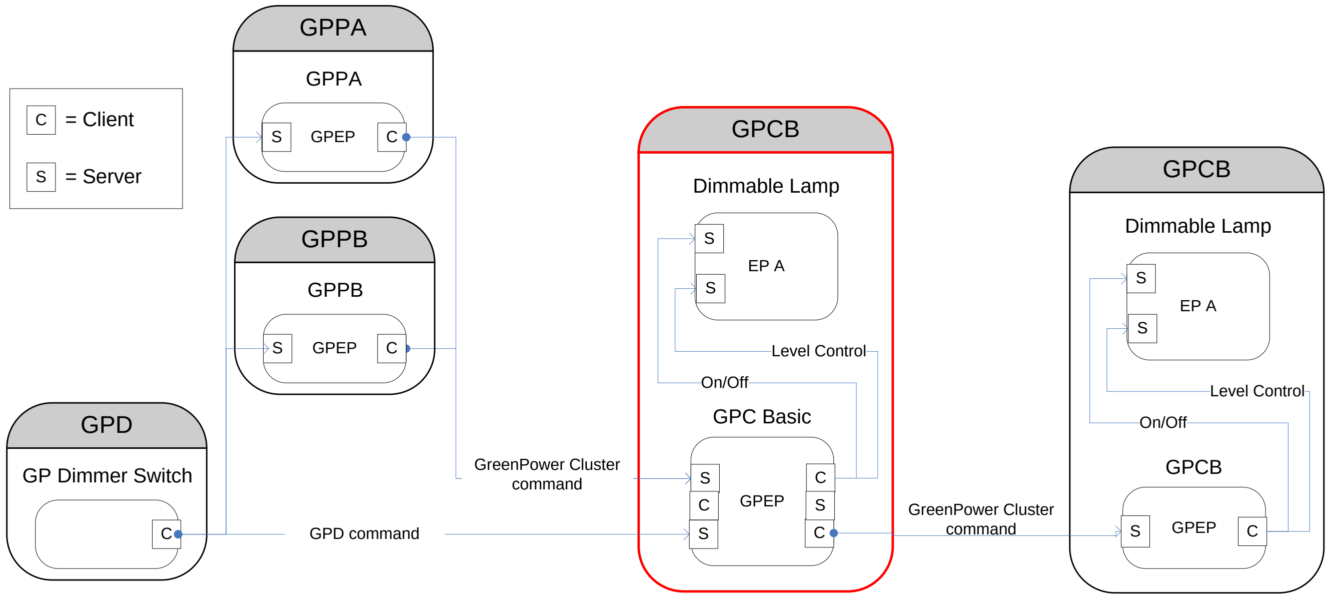


Figure 19 – Example of GP Combo Basic device usage

[51] CCB #2114; Resolution added in 15-02014-002

A.3.2.8 Proxy functionality

The GP specification defines various functionality block of Green Power protocol (see sec. A.3.3.2.7 and A.3.4.2.7).

Table 21 describes the proxy functionality. According to the current specification, only proxy functionality of a Green Power Basic Proxy, standalone or as part of Green Power Basic Combo, can be implemented. Other functionality and elements, intended for Advanced Proxy devices, are kept in for reference; where possible, they are indicated clearly.

Table 21 consists of three columns:

- The leftmost column contains the name of a functionality block;
- The middle column provides an overview of the GP objects (commands, attributes, primitives, functions, etc.) utilized by each functionality block in the proxy, and is informative, i.e. meant for implementation support only. The sections describing a particular functionality or object contain further implementation details (e.g. the M/O elements, or elements to be supported by basic/advanced proxies).
- The rightmost column is normative and indicates if a particular functionality block is mandatory/optional for a Green Power Basic Proxy, standalone or as part of Green Power Basic Combo.

Table 21 – Functionality of proxy device

Functionality	Elements in a proxy	M/O implementation for a Proxy Basic
Common elements	Green Power EndPoint duplicate filtering, , Proxy Table attribute, gppFunctionality, gppActiveFunctionality attribute, gppMaxProxyTable attribute, Rx GP Pairing, Rx Device_annce, Tx Device_annce for alias conflict, Rx GP Proxy Table Request, Tx GP Proxy Table Response	M
Direct communication (reception of GPDF via GP stub)	GP stub for GPDF reception (incl. GPD security), GP-SEC.request, GP-SEC.response	M
GPD IEEE address support	GPDF format, Proxy Table entry format, format of all proxy-supported Green Power cluster commands carrying GPDID	M
gpdSecurityLevel = 0b00	gpdSecurityLevel = 0b00 frame processing in the GP stub and Green Power EndPoint	M
gpdSecurityLevel = 0b10	gpdSecurityLevel = 0b10 frame processing in the GP stub and Green Power EndPoint	M
gpdSecurityLevel = 0b11	gpdSecurityLevel = 0b11 frame processing in the GP stub and Green Power EndPoint	M
Derived groupcast communication	Tx groupcast GP Notification to GPDID-derived GroupID with/without alias after Dmin/gppTunnelingDelay	M
Pre-commissioned groupcast communication	Tx groupcast GP Notification to a pre-configured GroupID, with/without alias, after Dmin/gppTunnelingDelay	M
Full unicast communication	gppTunnelingDelay, Tx GP Tunneling Stop with alias, Rx GP Tunneling Stop, drop own scheduled transmission on Rx GP Tunneling Stop, Tx unicast GP Notification without alias, Rx GP Notification Response, retry, *gppNotificationRetryNumber* and *gppNotificationRetryTimer* attribute	X
Lightweight unicast communication	Tx unicast GP Notification without alias after Dmin,	M

Multi-hop commissioning (unidirectional & bidirectional commissioning, with channel and shared key delivery over the air)	Rx GP Proxy Commissioning Mode, commissioning mode, Rx GP Response in commissioning, gpTxQueue, Maintenance GPDF format for Channel Request/Configuration, Tx GP Commissioning Notification in broadcast/unicast, with/without alias, after Dmin/gppTunnelingDelay, Advanced elements: *gpSharedSecurityKeyType* and *gpSharedSecurityKey* attribute, Rx GP Commissioning Notification, drop own scheduled transmission on Rx GP Commissioning Notification with better TempMaster	M
CT-based commissioning	Read access to Proxy Table, Write access to Proxy Table/Rx GP Pairing	M
Bidirectional communication in operational mode	GP stub for Tx (incl. security), gpTxQueue, gppTunnelingDelay, Tx GP Notification without alias, Rx GP Notification, drop own scheduled transmission on Rx GP Notification with better TempMaster, Rx GP Response in operation,	X
Proxy Table maintenance (for GPD mobility, proxy mobility and proxy link robustness)	Tx broadcast GP Notification, Tx GP Pairing Search, Rx GP Pairing, passive discovery, active discovery, Rx GP Notification, discover communication modes used; inactive/invalid Proxy Table entries; gppBlockedGPDID attribute, *gppMaxSearchCounter* attribute	X

A.3.2.9 Sink functionality

The GP specification defines various functionality block of Green Power protocol (see sec. A.3.3.2.7 and A.3.4.2.7).

Table 22 describes the [52]sink functionality. According to the current specification, only sink functionality of a Green Power Basic Sink, standalone or as part of Green Power Basic Combo, can be implemented. Other functionality and elements, intended for advanced sink devices, are kept in for reference; where possible, they are indicated clearly.

Table 22 consists of three columns:

- The leftmost column contains the name of a functionality block;
- The middle column provides an overview of the GP objects (commands, attributes, primitives, functions, etc.) utilized by each functionality block by the sink, and is informative, i.e. meant for implementation support only. The sections describing a particular functionality or object contain further implementation details (e.g. the M/O support of elements, or elements to be supported by basic/advanced sinks).
- The rightmost column is normative, and indicates if a particular functionality block is mandatory/optional for a Green Power Basic Sink, standalone or as part of Green Power Basic Combo.

Table 22 – Functionality of sink device

Functionality	Elements in a sink	Basic Sink
Common elements	GPEP duplicate filtering, Sink Table attribute, gpsMaxSinkTable attribute, gppFunctionality, gppActiveFunctionality attribute, GPD command translation, GPD command execution, Rx GP Sink Table Request, Tx GP Sink Table Response, gpsCommunicationMode attribute, gpsCommissioningExitMode attribute, gpsSecurityLevel attribute; shared security,	M
Direct communication (reception of GPDF via GP stub)	GP stub for GPDF reception (incl. security), GP-SEC.request, GP-SEC.response,	O
GPD IEEE address support	[53]The implementation of the GPD IEEE address support functionality does not mandate any new elements; however, it influences format of the following elements: GPDF format, Sink Table entry format, GPD Command Translation Table entry format (if supported), format of all sink-supported Green Power cluster commands carrying GPDID	M
gpdSecurityLevel = 0b00	gpdSecurityLevel = 0b00 frame processing in the GP stub (if direct communication supported) and Green Power EndPoint	O
gpdSecurityLevel = 0b10	gpdSecurityLevel = 0b10 frame processing in the GP stub (if direct communication supported) and Green Power EndPoint	M
gpdSecurityLevel = 0b11	gpdSecurityLevel = 0b11 frame processing in the GP stub (if direct communication supported) and Green Power EndPoint	M
Derived groupcast communication	Rx groupcast GP Notification with GPDID-derived GroupID	O.1
Pre-commissioned groupcast communication	Rx groupcast GP Notification with pre-configured GroupID, Tx GP Pairing Configuration	O.1 (M if derived groupcast supported)
Full unicast communication	Rx unicast GP Notification, Tx GP Notification Response	X
Lightweight unicast communication	proxy selection, Tx unicast GP Pairing, Rx unicast GP Notification,	O.1

[52] CCB #2417, resolved in 15-02014-010
[53] CCB #2326; Resolution added in 15-02014-011

Proximity commissioning (unidirectional & bidirectional, with channel and shared key delivery over the air)	Commissioning mode, Rx GPD Channel Request command, Tx GPD Channel Configuration command, Rx GPD Commissioning command, Tx GPD Commissioning Reply command, GPD application functionality matching, GPD security matching, Rx GPD Success command, Rx GPDF Success, Tx GP Pairing, Tx Device_annce for the alias (but NOT in the case of lightweight unicast communication), Rx GPD Decommissioning command, opt. Rx Sink Commissioning Mode command, O: Rx Data GPDF with *Auto-Commissioning* = 0b1. *gpSharedSecurityKeyType* attribute, *gpSharedSecurityKey* attribute TC-LK decryption of OOB key, M: TC-LK encryption of shared key **Proximity:** gpTxQueue, GP stub for GPDF reception, GP stub for GPDF transmission, Maintenance GPDF format for Channel Request/Configuration,	O (M if Direct communication supported)
Proxy-based commissioning (unidirectional & bidirectional, with channel and shared key delivery over the air)	Commissioning mode, Rx GPD Channel Request command, Tx GPD Channel Configuration command, Rx GPD Commissioning command, Tx GPD Commissioning Reply command, GPD application functionality matching, GPD security matching, Rx GPD Success command, Rx GPDF Success, Tx GP Pairing, Tx Device_annce for the alias (but NOT in the case of lightweight unicast communication), Rx GPD Decommissioning command, opt. Rx Sink Commissioning Mode command, O: Rx Data GPDF with *Auto-Commissioning* = 0b1. *gpSharedSecurityKeyType* attribute, *gpSharedSecurityKey* attribute TC-LK decryption of OOB key, M: TC-LK encryption of shared key **Proxy-based:** Tx GP Proxy Commissioning Mode, Rx GP Commissioning Notification in broadcast or unicast, Tx GP Response, Temp Master election,	M
CT-based commissioning	Read access to Sink Table, Write access to Sink Table/Rx GP Pairing Configuration, opt. Rx Sink Commissioning Mode command OPTIONAL: Translation Table, Rx Translation Table Update, Rx Translation Table Request, Tx Translation Table Response,	M
Proximity bidirectional communication in operational mode	GP stub for GPDF transmission (incl. security), gpTxQueue	X
Multi-hop bidirectional communication in operational mode	Rx GP Notification, TempMaster election, Tx GP Response in operation,	X
Maintenance of GPD (channel/key over-the-air update in operational mode)	Rx GP Notification , Temp Master election, Tx GP Response in operation, generate GPD Channel Configuration in operation, generate GPD Commissioning Reply command in operation	X
Proxy Table maintenance (for GPD mobility, GPP mobility and GPP robustness)	Rx broadcast GP Notification, Rx GP Pairing Search, Tx GP Pairing	X
Sink Table-based groupcast forwarding	Tx GP Pairing Configuration, Rx GP Pairing Configuration, Tx groupcast GP Notification OPTIONAL: Translation Table, Rx Translation Table Update, Rx Translation Table Request, Tx Translation Table Response	X
Translation Table	Translation Table, Rx Translation Table Update, Rx Translation Table Request, Tx Translation Table Response	O
[54]Compact attribute reporting	Rx Application Description GPDF, Rx Compact Attribute Reporting GPDF OPTIONAL: Translation Table, Rx Translation Table Update, Rx Translation Table Request, Tx Translation Table Response	O

A.3.2.10 GP command support per GP infrastructure device

Table 23 summarizes GP commands support required for each device type of GP infrastructure device. The following notations are used to indicate the requirement status:

- M Mandatory

[54] Dec 2016 SVE comment: https://workspace.zigbee.org/kws/groups/PRO_GP/comments/view_comment?comment_id=1014

• O	Optional
• O.n	Optional, but support of at least one of the group of options labeled O.n is required.
• N/A	Not applicable
• X	Prohibited

2698 **Table 23 – Green Power cluster: command implementation by GP infrastructure device**

Command Name	Implementation			
	Basic Proxy (standalone or of Basic Combo)		Sink side of in Basic Combo	
	Tx	Rx	Tx	Rx
GP Notification	Groupcast: M WithAlias: M FUnicast: X LUnicast: M WithoutAlias: M Broadcast: X	Groupcast: X FUnicast/LUnicast: N/A Broadcast: X	N/A	M (at least one of groupcast/FUnicast/LUnicast) Broadcast: X
GP Tunneling Stop	X	X	N/A	N/A
GP Pairing Search	X	X	N/A	X
GP Notification Response	N/A	X	X	N/A
GP Pairing	N/A	M	M	N/A
GP Proxy Commissioning Mode	N/A	M	M	O
GP Commissioning Notification	unicast: M broadcast: M	X	N/A	M (at least one of unicast and broadcast)
GP Response	N/A	In commissioning: M, In operation: X	In commissioning: M, In operation: X	In commissioning: M, In operation: X
GP Translation Table Update command	N/A	N/A	O	O
GP Translation Table Request	N/A	N/A	O	O
GP Translation Table Response	N/A	N/A	O	O

zigbee alliance

GP Pairing Configu-ration	N/A	N/A	O (M for sinks with *CommunicationMode*=0b10)	M
GP Sink Table Re-quest	O	N/A	N/A	M
GP Sink Table Re-sponse	N/A	O	M	N/A
GP Proxy Table Request	N/A	M	O	N/A
GP Proxy Table Response	M	N/A	N/A	O
GP Sink Commis-sioning Mode Com-mand	N/A	N/A	O	O

A GP infrastructure device SHALL silently drop any received GP command it does not support. Unless explicitly specified otherwise, it SHALL NOT send the ZCL Default Response command.

A.3.3 Server

A.3.3.1 Dependencies

None.

A.3.3.2 Server Attributes

The server side of the Green Power cluster contains the attributes shown in Table 24. The M/O column indicates if it is mandatory or optional to support this attribute.

Table 24 applies to sink devices.

Table 24 – Attributes of the GP server cluster

ID	Name	Type	Range	Access	Default	M/O	Description
0x0000	*gpsMaxSinkTableEntries*	unsigned 8-bit integer	Any valid	R	0x05 / 0x0a	M	Maximum number of Sink Table entries supported by this device
0x0001	*SinkTable*	Long octet string	N/A	R	0x0000	M	Sink Table, holding information about local bindings between a particular GPD and target's local endpoints
0x0002	*gpsCommunicationMode*	8-bit bitmap	N/A	R/W	0x01	M	Default communication mode requested by this sink
0x0003	*gpsCommissioningExitMode*	8-bit bitmap	N/A	R/W	0x02	M	Conditions for the sink to exit the commissioning mode
0x0004	*gpsCommissioningWindow*	unsigned 16-bit integer	Any valid	R/W	0x00B4	O	Default duration of the Commissioning window duration, in seconds, as requested by this sink
0x0005	*gpsSecurityLevel*	8-bit bitmap	N/A	R/W	0x06	M	The minimum required security level to be supported by the paired GPDs
0x0006	*gpsFunctionality*	24-bit bitmap	N/A	R	Any valid	M	The optional GP functionality supported by this sink
0x0007	*gpsActiveFunctionality*	24-bit bitmap	N/A	R	0xffffff	M	The optional GP functionality supported by this sink that is active
0x0008-0x000f	Reserved for other attributes of Green Power cluster server side						
0x0010-0x001f	Defined by the Client side (A.3.4.2)						
0x002-0x002f	Reserved for attributes shared by client and server side of the Green Power cluster (see Table 30)						
0x0030-0xffff	Reserved						

[55]With respect to ZCL Default Response handling for the ZCL foundation commands to manipulate the GP sink attributes, the sink SHALL follow section 2.5.12.2 of ZCL r06 or later (see [3]) and, in addition, for ZCL Write Attributes command, also section 2.5.3.3 of ZCL r06 or later (see [3]).

A.3.3.2.1 gpsMaxSinkTableEntries

The *gpsMaxSinkTableEntries* attribute is one octet in length, and it contains the maximum number of Sink Table entries that can be stored by this sink.

The value of 0xff indicates unspecified. The value of 0x00 indicates that Sink Table is not supported.

[55] CCB #2336: Resolution added in 15-02014-009

Any sink type supporting the Sink Table based groupcast forwarding functionality SHALL support at least 10 Sink Table entries. Any sink type not supporting the Sink Table based groupcast forwarding functionality SHALL support at least 5 Sink Table entries.

A.3.3.2.2 Sink Table

The *Sink Table* attribute contains the pairings configured for this sink.

Sink Table is a read-only attribute. Generic ZCL commands cannot be used to create/modify or remove *Sink Table* entries. If required, e.g. for CT-based commissioning, the GP Pairing Configuration command of the Green Power cluster can be used for that purpose.

A.3.3.2.2.1 Over the air transmission of Sink Table

When sent over the air in a ZCL command carrying the Sink Table attribute, it is represented as long octet string, which internally has the format of a sequence of octets. Thus, it contains the 2B length field of the Long octet string data format – defining the total length of the attribute and then the Sink Table entries itself, each of which is a sequence of octets, formatted as shown in Table 25. For each of the entries, the presence of the optional parameters is indicated by the corresponding flag in the *Options* or *Security Options* parameter:

- The *GPD ID* and the *Endpoint* parameter:
  - *ApplicationID* = 0b000 indicates the *GPD ID* parameter has the length of 4B and contains the SrcID; the *Endpoint* parameter is absent.
  - *ApplicationID* = 0b010 indicates the *GPD ID* parameter has the length of 8B and contains IEEE address; the *Endpoint* parameter is present.
  - All values of *ApplicationID* other than 0b000 and 0b010 are reserved in the current version of the Green Power cluster specification.
- The *Group list* parameter:
  - SHALL only be included if *CommunicationMode* sub-field of the *Options* parameter is set to 0b10;

    whereby the first octet indicates the number of entries in the list, and the entries of the list follow directly, formatted as specified in Table 26;
  - SHALL be completely omitted otherwise (i.e. even the length field SHALL be omitted);
- *GPD Assigned Alias* parameter SHALL be included if the *AssignedAlias* sub-field of the *Options* field is set to 0b1, otherwise it SHALL be omitted;
- the parameters *Security Options* and *GPD key* SHALL always all be included if the *SecurityUse* sub-field is set to 0b1 (irrespective of the key type in use); *SecurityUse* sub-field is set to 0b0, the parameters *Security Options*, and *GPD key* SHALL be omitted.
- *GPD security frame counter* parameter SHALL:
  - be present and carry the value of the *Security frame counter*, if:
    – *SecurityUse* = 0b1,
    – *SecurityUse* = 0b0 and *MAC sequence number capabilities* = 0b1;
  - be omitted if *SecurityUse* = 0b0 and *MAC sequence number capabilities* = 0b0.

The sink SHALL only respond with ZCL Read Attributes Response with Status = SUCCESS, if all configured Sink Table entries fit completely into a single response frame (without fragmentation or partitioning cluster usage). Otherwise, the sink SHALL respond with ZCL Read Attributes Response with Status = INSUFFICIENT_SPACE and no entries included (for the values of the Status codes see [3]).

A.3.3.2.2.2 Sink Table entry format

Implementers of this specification are free to implement the Sink Table in any manner that is convenient and efficient, as long as it represents the data in Table 25.

The Sink Table SHALL be persistently stored.

Table 25 – Format of entries in the Sink Table

Parameter name	Type	Range	Default	M / O	Description
Options	16-bit bitmap	Any valid	N/A	M	The options for the reception from this GPD
GPD ID	Unsigned 32-bit Integer/IEEE address	Any valid	N/A	M	ID of the paired GPD
Endpoint	Unsigned 8-bit integer	0x01 – 0xf0, 0xff	N/A	O (M if *ApplicationID* = 0b010)	Identifier of the logical device on an IEEE-addressed GPD
DeviceID	8-bit enumeration	Any valid (see)	N/A	M	The DeviceID for this GPD
Group list	Sequence of octets	Any valid	N/A	O (M if *CommunicationMode* = 0b10)	The 16-bit GroupID and alias for the group communication.
GPD Assigned Alias	Unsigned 16-bit integer	0x0001-0xfff7	N/A	O (M if *AssignedAlias* = 0b1)	The commissioned 16-bit ID to be used as alias for this GPD
Groupcast radius	Unsigned 8-bit integer	0x00 – 0xff	0xff	M	To limit the range of the groupcast
Security Options	8-bit bitmap	Any valid	N/A	O (M if *Security use* = 0b1)	The security options
GPD security frame counter	Unsigned 32-bit Integer	Any valid	0xffffffff	O (M if *Security use* = 0b1 or *Sequence number capabilities* = 0b1 and *Security use* = 0b0)	The incoming security frame counter for the GPD
GPD key	Security key	Any valid	N/A	O	The security key for the GPD. It MAY be skipped, if common/derivable key is used (as indicated in the [56]*Security Options* parameter)

A.3.3.2.2.2.1 Options parameter of the Sink Table

The *Options* parameter has the format as shown in Figure 20.

Bits: 0..2	3..4	5	6	7	8	9	10..15
ApplicationID	Communication mode	Sequence number capabilities	RxOnCapability	FixedLocation	AssignedAlias	Security use	Reserved

[56] CCB #2570; Resolution added in 16-02607-025

Figure 20 – Format of the Options parameter of the *Sink Table* attribute

The *ApplicationID* sub-field contains the information about the application used by the GPD. *ApplicationID* = 0b000 indicates the GPD ID parameter has the length of 4B and contains the GPD SrcID. *ApplicationID* = 0b010 indicates the GPD ID parameter has the length of 8B and contains the GPD IEEE address; the *Endpoint* parameter of 1B length is present. All values of *ApplicationID* other than 0b000 and 0b010 are reserved in the current version of the Green Power cluster specification.

The *CommunicationMode* sub-field contains the information about the accepted tunneling mode for this GPD. It can take the values as defined in Table 27.

The *Sequence number capabilities* sub-field contains the information on the sequence number capabilities of this GPD. It takes the values as defined in sec. A.4.2.1.1.2.

The *RxOnCapability* sub-field contains the information about reception capability on this GPD.

The *FixedLocation* sub-field contains information if the location of this GPD is expected to change.

The *AssignedAlias* sub-field, if set to 0b1, indicates that the assigned alias as stored in the *GPD Assigned Alias* parameter SHALL be used instead of the alias derived from the GPD ID (sec. A.3.6.3.3) in case of derived groupcast or full unicast communication. If set to 0b0, the derived alias is used (sec. A.3.6.3.3) for those communication modes.

The *Security use* sub-field, if set to 0b1, indicates that security-related parameters of the Sink Table entry are present.

A.3.3.2.2.2.2 Endpoint field

The *Endpoint* field SHALL be present if *ApplicationID* = 0b010. It then carries the identifier of the GPD endpoint, which jointly with the GPD IEEE address identifies a unique logical GPD device.
If *ApplicationID* = 0b000 the *Endpoint* field SHALL be absent.

The values 0xf1 - 0xfe are reserved for future use. The value 0x00 indicates application endpoint-independent communication and SHOULD be used e.g. for channel and key updates. The value 0xff indicates 'all endpoints'.

A.3.3.2.2.2.3 DeviceID parameter

The *DeviceID* parameter stores then the DeviceID of the paired GPD, as communicated/derived (see sec. A.3.6.2.1) during the pairing procedure.

A.3.3.2.2.2.4 Group list parameter

The *Group list* parameter stores the GroupID and the corresponding alias for groupcast communication. The entries in the *Group list* parameter SHALL be formatted as specified in Table 26.

Table 26 – Format of entries in the *Sink group list* parameter

Parameter name	Type	Description
Sink group	Unsigned 16-bit integer	The GroupID, either pre-commissioned or derived
Alias	Unsigned 16-bit integer	The Alias to be used jointly with this GroupID, either pre-commissioned or derived

If the *CommunicationMode* sub-field of the *Options* parameter is set to 0b10, the *Group list* SHOULD be present.

The *Alias* field of the *Group list* entry set to 0xffff indicates usage of derived alias for the *Sink group* in the same *Group list* entry.

The *Group list* parameter of each Sink Table entry SHOULD be able to store at least two group entries.

A.3.3.2.2.2.5 Groupcast radius parameter

The *Groupcast radius* contains the intended radius for the groupcast communication, in number of hops. The default value of 0x00 indicates undefined, i.e. twice the value of the nwkMaxDepth attribute of the NIB, as specified by [1].

If *Groupcast radius* parameter is set to a value 0x00 and another value is received, the new value SHALL be kept. If *Groupcast radius* parameter is set to a value other than 0x00 and a new value is received, the higher value SHALL be kept.

In the ZCL command carrying the Sink Table attribute, the *Groupcast radius* parameter SHALL always be present.

A.3.3.2.2.2.6 Security-related parameters

The *Security Options* parameter is formatted as shown in Figure 21. It is present if the *Security use* sub-field is set to 0b1.

Bits: 0-1	2-4	5-7
SecurityLevel	SecurityKeyType	Reserved

Figure 21 – Format of the Security Options parameter

[57]The *SecurityLevel* sub-field can take values as defined in Table 11 in section A.1.4.1.3 and the *SecurityKeyType* sub-field can take values as defined in Table 53 in section A.3.7.1.2.

If *SecurityLevel* is 0b00 or if the *SecurityKeyType* has value 0b011 ([58]NWK-key derived GPD group key), [59]0b010 (GP group key), 0b001 (NWK key) or 0b111 (derived individual GPD key), the *GPDkey* parameter MAY be omitted and the key MAY be stored in the *gpSharedSecurityKey* parameter instead. If *SecurityLevel* has value other than 0b00 and the *SecurityKeyType* has value 0b111 (derived individual GPD key), the *GPDkey* parameter MAY be omitted and the key MAY calculated on the fly, based on the value stored in the *gpSharedSecurityKey* parameter.

The *GPD security frame counter* parameter stores the last observed valid frame counter value for this GPD.

A.3.3.2.3 gpsCommunicationMode attribute

The *gpsCommunicationMode* attribute contains the communication mode required by this sink; the last two bits can take values as defined in Table 27.

Table 27 – Values of *gpsCommunicationMode* attribute[60]

Value	Description
0b00	Full unicast forwarding of the GP Notification command by proxies supporting the full unicast functionality (with observing of *gppTunnelingDelay* and with the transmission/reception of the GP Tunneling Stop command and with GP Notification retry when not receiving GP Notification Response); see sec. A.3.5.2.1
0b01	groupcast forwarding of the GP Notification command to DGroupID (see A.3.6.1.4); see sec. A.3.5.2.3
0b10	groupcast forwarding of the GP Notification command to pre-commissioned GroupID; see sec. A.3.5.2.3

[57] CCB #2292: Resolution added in 15-02014-006
[58] CCB #2565; Resolution added in 16-02607-025
[59] CCB #2565; Resolution added in 16-02607-025
[60] CCB #2276: Resolution added in 15-02014-006

Value	Description
0b11	unicast forwarding of the GP Notification command by proxies supporting the lightweight unicast functionality (i.e. without *gppTunnelingDelay* and without the transmission/reception of the GP Tunneling Stop command, and without GP Notification retry when not receiving GP Notification Response) ; see sec. A.3.5.2.3

If the *gpsCommunicationMode* has the value of 0b00 or 0b01, the mode 0b10 can be used instead for a pairing with particular GPD, if it is established so in the commissioning process.

If the *gpsCommunicationMode* value 0b11 is used, it is the responsibility of the sink (or commissioning tool, or another intelligent device in the network) to create the Proxy Table entries for the GPD on the required number of proxies, which implement lightweight unicast forwarding.

A.3.3.2.4 gpsCommissioningExitMode attribute

The *gpsCommissioningExitMode* attribute contains the information on commissioning mode exit requirements of this sink. It has the format as indicated in Figure 22.

Bits: 0	1	2	3..7
On CommissioningWindow expiration	On first Pairing success	On GP Proxy Commissioning Mode (exit)	Reserved

Figure 22 – Format of the *Commissioning Exit Mode* attribute

Only one of the flags *On GP Proxy Commissioning Mode (exit)* and *On first Pairing success* SHALL be set to 0b1 at the same time. The *On CommissioningWindow expiration* flag can be set to 0b1 in combination with any of the other flags or alone.

A.3.3.2.5 gpsCommissioningWindow attribute

The *gpsCommissioningWindow* attribute contains the information on the time, in seconds, during which this sink accepts pairing changes (additions/removals).

The default value is 180 seconds.

A.3.3.2.6 gpsSecurityLevel attribute

The *gpsSecurityLevel* attribute contains the minimum security level this sink requires the paired GPDs to support. It has the format as indicated in Figure 23.

Bits: 0-1	2	3	4..7
Minimal GPD Security Level	Protection with gpLinkKey	Involve TC	Reserved

Figure 23 – Format of the *gpsSecurityLevel* attribute

The *Minimal GPD Security Level* sub-field contains the minimum gpdSecurityLevel this sink accepts. It can take values as defined in Table 11.

The *Protection with the gpLinkKey* sub-field, indicates if the GPDs attempting the pairing are required to support protecting the over-the-air exchange of the GPD Key (as indicated by the *GPDkeyEncryption* sub-field of the *Extended Options* field of the GPD Commissioning command).

The *Involve TC* sub-field, if set to 0b1, overrides the settings of the *Minimal GPD Security Level* and the *Protection with the gpLinkKey* sub-fields. It indicates the sink SHALL NOT take the commissioning decisions on its own and SHALL contact the Trust Centre instead.
According to the current version of the specification, sinks joining a distributed Zigbee network or joining using the default Trust Centre Link Key SHALL set this bit to 0b0. Sinks joining the Zigbee network using IC-based unique link key SHALL set this bit to 0b1; since in the current version of the specification the mechanism to involve the TC in the GPD commissioning is not defined, if the *Involve TC* sub-field of the *gpsSecurityLevel* attribute is set to 0b1, the sink implemented according to the current specification SHALL NOT engage in GPD commissioning (see sec. A.3.9.1, step 1).

A TC or a CT MAY overwrite the setting of the *gpsSecurityLevel* attribute at any time.

The GP Pairing Configuration command, SHALL still be accepted on reception, as described in A.3.5.2.4.1, even if the *Involve TC* sub-field of the *gpsSecurityLevel* attribute is set to 0b1.

The attribute SHALL be persistently stored.

A.3.3.2.7 gpsFunctionality attribute

The *gpsFunctionality* attribute indicates support of the GP functionality by this device. Any 1-bit sub-field set to 0b1 indicates that this functionality is supported; set to 0b0 indicates that this functionality is not implemented.

The reserved sub-fields and sub-fields for any non-applicable functionality SHALL also be set to 0b0.

The *gpsFunctionality* attribute is formatted as shown in Table 28.

The rightmost column shows the values used by the Basic Sink, standalone or as part of Green Power Basic Combo.

Table 28 – Format of the gpsFunctionality attribute

Indication	Functionality	Basic Sink
b0	GP feature	0b1
b1	Direct communication (reception of GPDF via GP stub)	device-specific
b2	Derived groupcast communication	device-specific
b3	Pre-commissioned groupcast communication	device-specific
b4	Full unicast communication	0b0
b5	Lightweight unicast communication	device-specific
b6	Proximity bidirectional operation	0b0
b7	Multi-hop bidirectional operation	0b0
b8	Proxy Table maintenance (active and passive, for GPD mobility and proxy robustness)	0b0
b9	Proximity commissioning (unidirectional and bidirectional)	device-specific
b10	Multi-hop commissioning (unidirectional and bidirectional)	0b1
b11	CT-based commissioning	0b1
b12	Maintenance of GPD (deliver channel/key during operation)	0b0
b13	gpdSecurityLevel = 0b00 in operation	device-specific
b14	Deprecated: gpdSecurityLevel = 0b01	0b0
b15	gpdSecurityLevel = 0b10	0b1
b16	gpdSecurityLevel = 0b11	0b1
b17	Sink Table-based groupcast forwarding	0b0
b18	Translation Table	device-specific

b19	GPD IEEE address	0b1
[61]b20	Compact attribute reporting	device-specific
b21 – b23	Reserved	0b0

Note: the *gpdSecurityLevel* = 0b00 (bit 13) of the *gpsFunctionality* attribute encodes the device's support of unprotected GPDF in operation. During commissioning, it is mandatory for GP infrastructure devices to support the exchange of the GPD commands Channel Request, Channel Configuration, Commissioning and Commissioning Reply with *gpdSecurityLevel* = 0b00; therefore there is no need to encode that on bit 13 of the *gpsFunctionality*.

A.3.3.2.8 gpsActiveFunctionality attribute

The *gpsActiveFunctionality* attribute indicates which GP functionality supported by this device is currently enabled. Any 1-bit sub-field set to 0b1 indicates that this functionality is supported and enabled; set to 0b0 indicates that this functionality is disabled or not implemented.

The *gpsActiveFunctionality* attribute is formatted as shown in

[61] Dec 2016 SVE comment: https://workspace.zigbee.org/kws/groups/PRO_GP/comments/view_comment?comment_id=1014

Table 29.

Table 29 – Format of the gpsActiveFunctionality attribute

Indication	Functionality
b0	GP functionality
b1 – b23	Set to fixed value 0b1 in this specification.

The *GP feature* sub-field on b0 of the *gpsActiveFunctionality* attribute is a master flag. By writing 0b1/0b0 to the *GP feature* sub-field, the complete GP operation can be enabled/disabled, respectively. Even when the *GP feature* sub-field is set to 0b0, the GP attributes SHALL be accessible and the Simple Descriptor for the Green Power EndPoint SHALL still be readable.

In the current version of the GP specification, the *gpsActiveFunctionality* attribute is read only, and the *GP feature* sub-field SHALL be set to 0b1.

In the current version of the GP specification, the remaining sub-fields of the *gpsActiveFunctionality* attribute are reserved and SHALL be set to 0b1. If future version of the GP specification would define further *gpsActiveFunctionality* flags, they SHOULD be aligned with *gpsFunctionality* attribute.

A.3.3.3 Attributes shared by client and server

Both server and client side of the Green Power cluster contain the attributes shown in Table 30. The M/O column indicates if it is mandatory or optional to support this attribute.

Table 30 – Attributes shared by client and server of the Green Power cluster

ID	Name	Type	Range	Access	Default	M/O	Description
0x0020	*gpSharedSecurityKeyType*	8-bit bitmap	0x00-0x07	R/W	0b000	Basic Proxy: O Basic Sink: M	The security key type to be used for the communication with all paired GPD in this network
0x0021	*gpSharedSecurityKey*	128-bit security key	Any valid	R/W	N/A	Basic Proxy: O Basic Sink: M	The security key to be used for the communication with all paired GPD in this network
0x0022	*gpLinkKey*	128-bit security key	Any valid	R/W	'ZigbeeAlliance09'	M	The security key to be used to encrypt the key exchanged with the GPD
0x0023-0x002f	Reserved for other attributes shared by sink and proxy						
[62]0xfffd	*ClusterRevision*	Unsigned 16-bit integer	Any valid	R	0x0002	M	See ZCL [3]

A.3.3.3.1 gpSharedSecurityKeyType

The *gpSharedSecurityKeyType* attribute stores the key type of the shared security key. The *gpSharedSecurityKeyType* attribute can take the following values from Table 53: 0b000 (no key), 0b001 (NWK key), 0b010 (GP group key), 0b011 (NWK-key derived GP group key) and 0b111 (Derived individual GPD key).

[62] PoC comment #23 (Zigbee document 16-02601)

A.3.3.3.2 gpSharedSecurityKey

The *gpSharedSecurityKey* attribute stores the shared security key of the key type as indicated in the *gpSecurityKeyType* attribute. It can take any value.

If the *gpSharedSecurityKeyType* attribute has the value of 0b010 or 0b111, the *gpSharedSecurityKey* SHALL store the GP group key.
If the *gpSharedSecurityKeyType* attribute has the value of 0b000, 0b001 and 0b011, storing of the *gpSharedSecurityKey* MAY be omitted and writing to the *gpSharedSecurityKey* attribute has no effect.
If the *gpSharedSecurityKeyType* attribute has the value of 0b001, the *gpSharedSecurityKey* can be retrieved from the NIB *nwkSecurityMaterialSet* attribute.

A.3.3.3.3 gpLinkKey

The *gpLinkKey* attribute stores the Link Key, used to encrypt the key transmitted in the Commissioning GPDF and Commissioning Reply GPDF.

By default, it has the value of the default Zigbee Trust Center Link Key (TC-LK), ‘ZigbeeAlliance09’. Then, storing of the *gpLinkKey* MAY be omitted.

Note: change of the value of the *gpLinkKey* attribute SHALL NOT change the value of the Zigbee TC-LK.

A.3.3.4 Commands received

The cluster specific commands received by the server side of the GP cluster are listed in Table 31.

Whether the support of particular command is mandatory or optional is dependent on the GP infrastructure device type and the features it supports, and specified in Table 23.

Table 31 – Green Power cluster: server side: commands received

Command ID	Command Name	Command Description	Link
0x00	GP Notification	From proxy to sink to tunnel GP frame.	A.3.3.4.1
0x01	GP Pairing Search	From proxy to the sinks in entire network to get pairing indication related to GPD for Proxy Table update	A.3.3.4.2
0x02	Reserved		
0x03	GP Tunneling Stop	From proxy to neighbor proxies to indicate GP Notification sent in full unicast mode.	A.3.4.4.1
0x04	GP Commissioning Notification	From proxy to sink to tunnel GPD commissioning data.	A.3.3.4.3
0x05	GP Sink Commissioning Mode	To enable commissioning mode of the sink, over the air	A.3.3.4.8
0x06	Reserved		
0x07	GP Translation Table Update command	To configure GPD Command Translation Table	A.3.3.4.4
0x08	GP Translation Table Request	To provide GPD Command Translation Table content	A.3.3.4.5
0x09	GP Pairing Configuration	To configure Sink Table	A.3.3.4.6
0x0a	GP Sink Table Request	To read out selected Sink Table entries, by index or by GPD ID	A.3.3.4.7
0x0b	GP Proxy Table Response	To receive information on requested selected Proxy Table entries, by index or by GPD ID	A.3.4.4.2

0x0c-0xff	Reserved		

A.3.3.4.1 GP Notification command

The payload of the GP Notification command SHALL be formatted as illustrated in Figure 24.

Octets	2	4/8	0/1	4	1	1/variable	0/2	0/1
Data Type	16-bit bitmap	unsigned 32-bit integer/IEEE address	Unsigned 8-bit integer	unsigned 32-bit integer	unsigned 8-bit integer	Octet string	unsigned 16-bit integer	8-bit bitmap
Field Name	Options	GPD ID	Endpoint	GPD security frame counter	GPD CommandID	GPD Command payload	GPP short address	GPP-GPD link

Figure 24 – Format of the GP Notification command

Bits: 0..2	3	4	5	6-7	8-10
ApplicationID	Also Unicast	Also Derived Group	Also Commissioned Group	SecurityLevel	SecurityKeyType

Figure 25 – Format of the Options field of the GP Notification command (part 1)

Bits: 11	12	13	14	15
RxAfterTx	gpTxQueueFull	Bidirectional capability	ProxyInfoPresent	Reserved

Figure 26 – Format of the Options field of the GP Notification command (part 2)

The *ApplicationID* sub-field contains the information about the application used by the GPD. *ApplicationID* = 0b000 indicates the GPD ID field has the length of 4B and contains the GPD SrcID; the *Endpoint* field is absent. *ApplicationID* = 0b010 indicates the GPD ID field has the length of 8B and contains the GPD IEEE address; the *Endpoint* field is present. All values of *ApplicationID* other than 0b000 and 0b010 are reserved in the current version of the Green Power cluster specification.

The flags *Also Unicast*, *Also Derived Group* and *Also Commissioned Group* indicate presence of the sinks paired to the same GPD with a different communication mode, as stored in this proxy's Proxy Table.

The *SecurityLevel* sub-field has value copied from the received GPDF and can take values as specified in Table 11.

The *SecurityKeyType* sub-field has the value corresponding to the type of the key successfully used for security processing of the received GPDF, and can take values as specified in Table 53.

The *RxAfterTx* sub-field SHALL be copied from the *RxAfterTx* sub-field of the *Extended NWK Frame Control* field of the triggering GPDF was set; irrespective of bidirectional communication capabilities of the device sending the GP Notification.

The *gpTxQueueFull* sub-field indicates whether the proxy can still receive and store a GPDF Response for this GPD. If this field value is 0b0, there is space in the gpTxQueue for this GPD. If this field is set to 0b1, there is no space left in the gpTxQueue for this GPD. A forwarding device not supporting bidirectional communication SHALL always set this field to 0b1.

The *BidirectionalCommunicationCapability* sub-field, when set to 0b0, indicates that the device sending the GP Notification command does NOT support bidirectional communication. All proxy basic devices implementing the current specification SHALL always set the *BidirectionalCommunicationCapability* sub-field to 0b0.

The *ProxyInfoPresent* sub-field, when set to 0b1, indicates that the fields *GPP short address* and *GPP-GPD link* fields are present. All proxy basic device implementing the current specification SHALL always set *ProxyInfoPresent* sub-field to 0b1.

Note for sink implementers: Proxy devices implementing earlier versions of the Green Power specification will set the ProxyInfoPresent *sub-field to 0b0, and the optional presence of the proxy-related fields in the GP Commissioning Notification command will be indicated by its* RxAfterTx *sub-field of the* Options *field set to 0b1. In that case, the last octet of the proxy information will carry instead of the 8-bit bitmap* GPP-GPD link *value, a uint8* Distance *value (the higher the value, the worse the link). If and how the sinks use that legacy information, is application-specific and out of scope for the current specification.*

The *GPD ID* field has the value copied from the GPDF *SrcID*/GPDF MAC *Source address* field, depending on the *ApplicationID* sub-field value in the GPDF.

The *Endpoint* field, if *ApplicationID* = 0b010, is present and carries the value copied from the *Endpoint* field of the GPDF.

The *GPD security frame counter* field is always present. If the *SecurityLevel* sub-field of the *Extended NWK Frame Control* field of the received GPDF was 0b00, it carries the value copied from the GPDF MAC header *Sequence number* field, pre-padded with 0x000000. Otherwise, if the *SecurityLevel* sub-field of the *Extended NWK Frame Control* field of the received GPDF was 0b10- 0b11, it carries the value copied from the *Security frame counter* field of the received GPDF that was successfully used for the security processing.

The *GPD CommandID* has the value copied from the GPDF GPD CommandID field.

The *GPD Command Payload* field is an octet string. The first octet contains the payload length, the following octets – the payload of the GPDF Command, copied from the GPDF Command payload field. The default value of 0xff indicates unspecified/no payload; 0x00 indicates no payload.

The *GPP short address* field, if present, carries the short address of the device originating the GP Notification.

The *GPP-GPD link* field, if present, indicates the quality of the received GPDF, as reported by the dGP-DATA.indication primitive.

The *GPP-GPD link* field of the GP Notification command is formatted as shown in Figure 27.

Bits: 0..5	6..7
RSSI	Link quality

Figure 27 – Format of the *GPP-GPD link* field of the GP Notification command

The *RSSI* sub-field of the *GPP-GPD link* field encodes the RSSI from the range <+8 ; -109> [dBm], with 2dBm granularity. It SHALL be calculated as follows:

- The RSSI parameter value as supplied by the dGP-DATA.indication primitive SHALL be capped to the range <+8 ; -109> [dBm],
  i.e. any value higher than +8dBm is represented as +8 dBm; any value lower than -109dBm is represented as -109dBm, the values within the range remain unmodified;
- 110 SHALL be added to the capped RSSI value, to obtain a non-negative value;
- The obtained non-negative RSSI value SHALL be divided by 2.

The *Link quality* sub-field of the *GPP-GPD link* field encodes the quality of the link between the GPD and the forwarding proxy, as defined in Table 32. Its calculation is vendor-specific and may be based e.g. on LQI or correlation value.

Table 32 – Values of the *Link quality* sub-field of the *GPP-GPD link* field

Value	Description
0b00	Poor
0b01	Moderate
0b10	High
0b11	Excellent

A.3.3.4.1.1 When generated

The GP Notification command is generated by the proxy (or a sink capable of Sink Table-based forwarding) to forward the received Data GPDF to the paired sinks.

A.3.3.4.1.2 Effect on Receipt

On receipt of the GP Notification command, a device is informed about a GPDF forwarded by a proxy. Also the device which received this frame is informed of bidirectional communication capability of the sender.

A.3.3.4.2 GP Pairing Search command

The payload of the GP Pairing Search command SHALL be formatted as illustrated in Figure 28.

Octets	2	4/8	0/1
Data Type	16-bit bitmap	unsigned 32-bit integer/IEEE address	Unsigned 8-bit integer
Field Name	Options	GPD ID	Endpoint

Figure 28 – Format of the GP Pairing Search command

The *Options* field of the GP Pairing Search command is formatted as shown in Figure 29.

Bits: 0..2	3	4	5	6	7	8..15
ApplicationID	Request Unicast Sinks	Request Derived Groupcast Sinks	Request Commis-sioned groupcast sinks	Request GPD Security Frame Counter	Request GPD Security key	Reserved

Figure 29 – Format of the Options field of the GP Pairing Search command

The *ApplicationID* sub-field contains the information about the application used by the GPD. *ApplicationID* = 0b000 indicates the GPD ID field has the length of 4B and contains the GPD SrcID; the *Endpoint* field is absent. *ApplicationID* = 0b010 indicates the GPD ID field has the length of 8B and contains the GPD IEEE address; the *Endpoint* field is present. All values of *ApplicationID* other than 0b000 and 0b010 are reserved in the current version of the Green Power cluster specification.

The *RequestUnicastSinks* sub-field SHALL be set to 0b1, if the proxy requests pairing information on full and lightweight unicast sinks for the GPD specified in *GPD ID* field, and – if *ApplicationID* = 0b010 – *Endpoint* field.

The *RequestDerivedGroupcastSinks* sub-field SHALL be set, if the proxy requests pairing information on sinks accepting derived groupcast communication mode for the GPD specified in *GPD ID* field.

The *RequestCommissionedGroupcastSinks* sub-field SHALL be set, if the proxy requests pairing information on sinks accepting pre-commissioned GroupID communication mode for the GPD specified in *GPD ID* field.

Using the flags *Request GPD Security key* and *Request GPD Security frame counter*, the proxy can request those security parameters for the GPD specified in *GPD ID* field.

The *GPD ID* field carries the value of the *GPD ID*, either GPD SrcID or GPD IEEE address, depending on the value of the *ApplicationID*, on which the information is requested.

The *Endpoint* field, if *ApplicationID* = 0b010, is present and carries the identifier of the GPD endpoint of an IEEE-addressed GPD, on which the information is requested.

The *Disable default response* sub-field of the *Frame Control Field* of the ZCL header SHALL be set to 0b1.

A.3.3.4.2.1 When generated

The GP Pairing Search command is generated when the proxy needs to discover pairing information for a particular GPD.

A.3.3.4.2.2 Effect on Receipt

On receipt of this command, the device is informed about a proxy requesting pairing information on particular GPD.

A.3.3.4.3 GP Commissioning Notification command

The payload of the GP Commissioning Notification command SHALL be formatted as illustrated in Figure 30.

Octets	2	4/8	0/1	4	1	1/variable	0/2	0/1	0/4
Data Type	16-bit bitmap	unsigned 32-bit integer/IEEE address	Unsigned 8-bit integer	Unsigned 32-bit integer	unsigned 8-bit integer	Octet string	Unsigned 16-bit integer	8-bit bitmap	Unsigned 32-bit integer
Field Name	Options	GPD ID	Endpoint	GPD security frame counter	GPD CommandID	GPD Command payload	GPP short address	GPP-GPD link	MIC

Figure 30 – Format of the GP Commissioning Notification command

The *Options* field of the GP Commissioning Notification command SHALL be formatted as shown in Figure 31.

Bits: 0..2	3	4..5	6..8	9	10	11	12..15
ApplicationID	RxAfterTx	SecurityLevel	SecurityKeyType	SecurityProcessingFailed	Bidirectional Capability	ProxyInfoPresent	Reserved

Figure 31 – Format of the Options field of the GP Commissioning Notification command

The *ApplicationID* sub-field contains the information about the application used by the GPD. *ApplicationID* = 0b000 indicates the GPD ID field has the length of 4B and contains the GPD SrcID; the *Endpoint* field is absent. *ApplicationID* = 0b010 indicates the GPD ID field has the length of 8B and contains the GPD IEEE address; the *Endpoint* field is present. All values of *ApplicationID* other than 0b000 and 0b010 are reserved in the current version of the Green Power cluster specification.

The *RxAfterTx* sub-field SHALL be copied from the *RxAfterTx* sub-field of the *Extended NWK Frame Control* field of the triggering GPDF was set; irrespective of bidirectional communication capabilities of the device sending the GP Commissioning Notification.

SecurityLevel is copied from the *SecurityLevel* sub-field of the *Extended NWK Frame Control* field of the received GPDF, also in the case when security check failed and the *SecurityProcessingFailed* sub-field is set to 0b1. If the *Extended NWK Frame Control* field is not present in the received GPDF, the *SecurityLevel* sub-field is set to 0b00.

SecurityKeyType corresponds to the type of the key successfully used for GPDF processing. When security check failed [63]or could not be performed due to lack of security parameters for this GPD and the *SecurityProcessingFailed* sub-field is set to 0b1, the *SecurityKeyType* sub-field SHALL be set to 0b000 if the *SecurityKey* sub-field of the *Extended NWK Frame Control* field of the received GPDF was set to 0b0, or to 0b100 if the *SecurityKey* sub-field of the *Extended NWK Frame Control* field of the received GPDF was set to 0b1. If the *Extended NWK Frame Control* field is not present in the received GPDF, the *SecurityKeyType* sub-field is set to 0b000.

SecurityProcessingFailed sub-field SHALL be set to 0b1, if the Commissioning GPDF was protected, but the security check failed [64]or could not be performed due to lack of security parameters for this GPD.

The *BidirectionalCommunicationCapability* sub-field, when set to 0b0, indicates that the device sending the GPD Commissioning Notification command does NOT support bidirectional communication. All proxy basic devices implementing the current specification SHALL always set the *BidirectionalCommunicationCapability* sub-field to 0b0.

The *ProxyInfoPresent* sub-field, when set to 0b1, indicates that the fields *GPP short address* and *GPP-GPD link* fields are present. All proxy basic device implementing the current specification SHALL always set *ProxyInfoPresent* sub-field to 0b1.

Note for sink implementers: Proxy devices implementing earlier versions of the Green Power specification will set the ProxyInfoPresent *sub-field to 0b0, and the optional presence of the proxy-related fields in the GP Commissioning Notification command will be indicated by its* RxAfterTx *sub-field of the* Options *field set to 0b1. In that case, the last octet of the proxy information will carry instead of the 8-bit bitmap* GPP-GPD link *value, a uint8* Distance *value (the higher the value, the worse the link). If and how the sinks use that legacy information, is application-specific and out of scope for the current specification.*

The *GPD ID* field has the value copied from the GPDF *SrcID* field/MAC header *Source address* field, depending on the value of the *ApplicationID* sub-field in the GPDF. If the GPD command was received with the Maintenance *Frame Type*, the *ApplicationID* sub-field of the *Options* field SHALL be set to 0b000 and the *GPD ID* SHALL carry the value 0x00000000.

The *Endpoint* field, if *ApplicationID* = 0b010, is present and carries the value copied from the *Endpoint* field of the commissioning GPDF.

[63] CCB #2362; Resolution added in GP Basic spec errata 15-02014-011
[64] CCB #2362; Resolution added in GP Basic spec errata 15-02014-011

The *GPD security frame counter* field is always present. If the *SecurityLevel* sub-field of the *Extended NWK Frame Control* field of the received GPDF was 0b00, it carries the value copied from the GPDF MAC header *Sequence number* field, pre-padded with 0x000000. Otherwise, if the *SecurityLevel* sub-field of the *Extended NWK Frame Control* field of the received GPDF was 0b10- 0b11 and *SecurityProcessingFailed* sub-field is set to 0b0, it carries the value copied from the *Security frame counter* field of the received GPDF that was successfully used for the security processing of the received GPDF; if the *SecurityLevel* sub-field of the *Extended NWK Frame Control* field of the received GPDF was 0b10- 0b11 and *SecurityProcessingFailed* sub-field is set to 0b1, it carries the value copied from the *Security frame counter* field of the received GPDF.

The GPD CommandID carries the GPD CommandID.

The *GPD Command Payload* field is an octet string. The first octet contains the payload length, the following octets – the payload of the GPDF Command, copied from the GPDF Command payload field. The default value of 0xff indicates unspecified/no payload; 0x00 indicates no payload.

If the *SecurityLevel* sub-field of the *Options* field is set 0b00 or 0b10 or if *SecurityLevel* sub-field of the *Options* field is set to 0b11 and the *SecurityProcessingFailed* sub-field of the *Options* field is set 0b1, the value *GPD CommandID* and *GPD Command Payload* is copied from the GPDF. If the *SecurityLevel* sub-field of the *Options* field is set to 0b11 and the *SecurityProcessingFailed* sub-field of the *Options* field is set 0b0, the *GPD CommandID* and *GPD Command Payload* carry the result of the successful decryption of the corresponding GPDF fields.

The *GPP short address* field, if present, carries the short address of the device originating the GP Notification.

The *GPP-GPD link* field, if present, indicates the quality of the received GPDF, as reported by the dGP-DATA.indication primitive. The *GPP-GPD link* field of the GP Commissioning Notification command is formatted as shown in Figure 27 and calculated as defined in sec. A.3.3.4.1.

The *MIC* field SHALL only be present if the *SecurityProcessingFailed* sub-field is set to 0b1.

A.3.3.4.3.1 When generated

The GP Commissioning Notification command is used by the proxy in commissioning mode to forward commissioning data to the sink(s).

A.3.3.4.3.2 Effect on Receipt

On receipt of the GP Commissioning Notification command, a device is informed about a GPD device seeking to manage a pairing.

Also the device which received this frame is informed of bidirectional commissioning capability of the sender.

A.3.3.4.4 [65]GP Translation Table Update command

The GP Translation Table Update command allows for creation and modification and/or removal of entries in the *GPD Command Translation Table* (see Table 48). The payload of the GP Translation Table Update command SHALL be formatted as illustrated in Figure 32.

[65] PoC comment #9 (Zigbee document 16-02601)

Octets	2	4/8	0/1	Variable	…	Variable
Data Type	16-bit bitmap	unsigned 32-bit integer/IEEE address	Unsigned 8-bit integer	Variable	…	Variable
Field Name	Options	GPD ID	GPD Endpoint	Translation 1	…	Translation N

Figure 32 – Format of the GP Translation Table Update command

The *Options* field of the GP Translation Table Update command SHALL be formatted as illustrated in Figure 33.

Bits: 0..2	**3..4**	**5..7**	**8**	**9..15**
ApplicationID	Action	Number of Translations	Additional information block present	Reserved

Figure 33 – Format of the Options field of the GP Translation Table Update command

The *ApplicationID* sub-field contains the information about the application used by the GPD. *ApplicationID* = 0b000 indicates the GPD ID field has the length of 4B and contains the GPD SrcID; the *GPD Endpoint* field is absent. *ApplicationID* = 0b010 indicates the GPD ID field has the length of 8B and contains the GPD IEEE address; the *GPD Endpoint* field is present. All values of *ApplicationID* other than 0b000 and 0b010 are reserved in the current version of the Green Power cluster specification.

The *Action* sub-field of the *Options* field can take the values as specified in Table 33.

Table 33 – Values of the *Action* sub-field of the *Option* field

Value	**Description**
0b00	Add Translation Table entry
0b01	Replace Translation Table entry
0b10	Remove Translation Table entry
0b11	Reserved

If the *Action* sub-field of the *Options* field is set to 0b00, each translation included in the GP Translation Table Update command is to be stored in the GPD Command Translation Table at the sink, in the entry number as specified by the *Index* field if that entry is empty. If the entry specified by the *Index* is not empty, the action SHALL NOT be executed; a ZCL Default Response command with status FAILURE (see [3]) MAY be returned. If the *Index* field has the value of 0xff, the sink SHALL choose any free entry. Already existing translation entry for the same (GPD ID, GPD Endpoint, GPD CommandID, EndPoint, Profile, Cluster) quintuple present in the sink's Command Translation Table, if any, SHALL NOT be affected. [66]In the current version of the specification, the *Index* field SHALL always be set to 0xff upon transmission and ignored upon reception. [67]Thus, if a sink implemented according to the current specification receives a Translation Table Update command with Index NOT equal to 0xFF, it SHALL process it as if the *Index* was set to 0xFF.

[66] October PoC comment #961: https://workspace.zigbee.org/kws/groups/PRO_GP/comments/view_comment?comment_id=961
[67] Kavi comment #2107 from the GP vScVE November 2018; resolution added in 16-02607-026

If the *Action* sub-field of the *Options* field is set to 0b01, each translation included in the GP Translation Table Update command is to be stored to the GPD Command Translation Table at the sink, in the entry number as specified by the *Index* field. Translation entry(s) for the same (GPD ID, GPD Endpoint, GPD CommandID, EndPoint, Profile, Cluster) quintuple stored in the sink's Command Translation Table under different *Index* value, if any, SHALL NOT be affected by this command. [68]In the current version of the specification, the *Index* field SHALL always be set to 0xff upon transmission and ignored upon reception. [69]If a sink implemented according to the current specification receives a Translation Table Update command with Index NOT equal to 0xFF, it SHALL process it as if the *Index* was set to 0xFF. Thus, effectively, in the current version of the specification, GP Translation Table Update command with *Action* = 0b01 results in the sink replacing any number of translation entry(s) for the same (GPD ID, GPD Endpoint, GPD CommandID, EndPoint, Profile, Cluster) quintuple by the supplied number of entries.

If the *Action* sub-field of the *Options* field is set to 0b10, each translation in the GP Translation Table Update command, as defined by the *Index* value, SHALL be removed from the GPD Command Translation Table at the sink. The values of the remaining sub-fields of the Translation field are ignored. If the *Index* field is set to 0xff, all entries for [70]the same (GPD ID, GPD Endpoint, GPD CommandID, EndPoint, Profile, Cluster) quintuple SHALL be removed; the remaining sub-fields of the *Translation* field SHALL then be ignored upon reception and can be set to any value upon transmission; the *Additional Information* field SHOULD NOT be included. [71]In the current version of the specification, the *Index* field SHALL always be set to 0xff upon transmission and ignored upon reception. [72]Thus, if a sink implemented according to the current specification receives a Translation Table Update command with Index NOT equal to 0xFF, it SHALL process it as if the *Index* was set to 0xFF.

The *Number of Translations* indicates how many Translation fields are included in the command. 0b000 indicates none.

The *Additional information block present* sub-field, if set to 0b1, indicates that the *Additional information block* field is present; if set to 0b0, it indicates that the *Additional information block* field is absent.

[73]If in the received GP Translation Table Update command, the *Contact bitmask* field of the *Additional Information* field for a GPD 8-bit vector: press [74]or a GPD 8-bit vector: release command is set to 0x00 or the *EndPoint* field set to 0xfc, but the sink does not support GPD processing in the application, the sink SHOULD [75] drop the frame and SHOULD respond to the originator with ZCL Default Response carrying Status = FAILURE.

The *GPD ID* field has the format of GPD *SrcID* /GPD *IEEE address*, depending on the value of the *ApplicationID* sub-field, and contains the identifier of the GPD for which the translations are being updated.

The *GPD Endpoint* field, if *ApplicationID* = 0b010, is present and carries the identifier of the GPD endpoint on an IEEE-addressed GPD for which the translations are being updated.

[68] October PoC comment #961: https://workspace.zigbee.org/kws/groups/PRO_GP/comments/view_comment?comment_id=961
[69] Kavi comment #2107 from the GP vScVE November 2018; resolution added in 16-02607-026
[70] October PoC comment #960: https://workspace.zigbee.org/kws/groups/PRO_GP/comments/view_comment?comment_id=960
[71] October PoC comment #961: https://workspace.zigbee.org/kws/groups/PRO_GP/comments/view_comment?comment_id=961
[72] Kavi comment #2107 from the GP vScVE November 2018; resolution added in 16-02607-026
[73] Clarification for a special case of Translation Table entry with Additional Information for GPD 8-bit vector: press command with *Contact bitmask* = 0x00, as agreed during GP WG call of November 16th, 2016
[74] GP multi-sensor LB v0.7 comment #972: https://workspace.zigbee.org/kws/groups/PRO_GP/comments/view_comment?comment_id=972
[75] GP multi-sensor LB v0.7 comment #972: https://workspace.zigbee.org/kws/groups/PRO_GP/comments/view_comment?comment_id=972

The *Translation* field of the GP Translation Table Update command is formatted as illustrated in Figure 34 and Figure 35.

Octets	1	1	1	2	2
Data Type	unsigned 8-bit integer	unsigned 8-bit integer	unsigned 8-bit integer	unsigned 16-bit integer	unsigned 16-bit integer
Field Name	Index	GPD Command ID	EndPoint	Profile	Cluster

Figure 34 – Format of the Translation field of the GP Translation Table Update command (part 1)

1	1	0/Variable	0/Variable
unsigned 8-bit integer	unsigned 8-bit integer	sequence of unsigned 8-bit integer	sequence of unsigned 8-bit integer
Zigbee Command ID	Zigbee Command payload length	Zigbee Command payload	Additional information block

Figure 35 – Format of the Translation field of the GP Translation Table Update command (part 2)

The *Index* field determines the Translation Table entry. The first entry has the *Index* value of 0. [76]In the current version of the specification, the *Index* field SHALL always be set to 0xff [77]upon transmission and ignored upon reception. [78]Thus, if a sink implemented according to the current specification receives a Translation Table Update command with Index NOT equal to 0xFF, it SHALL process it as if the *Index* was set to 0xFF.

The *EndPoint* field carries the endpoint for which this translation is valid. If it is set to any of the unreserved values (0x01-0xf0), the value can be used directly. If the *Endpoint* field is set to 0xff, the translation applies to all matching endpoints. If the *Endpoint* field is set to 0xfe, the endpoints to which this translation applies are to be derived by the sink itself. If the *Endpoint* field is set to 0xfd, the list of endpoints to which this translation applies remains unmodified.

If the *Cluster* field is set to 0xffff, the ClusterID from the triggering GPD command is to be used.

The *Zigbee Command payload length* field indicates the length of the *Zigbee Command payload* field. If the *Zigbee Command payload length* field is set to 0x00, there is no payload. If the *Zigbee Command payload length* field is set to 0xff, the payload from the triggering GPD command is to be used. [79]If the *Length* sub-field of the *Zigbee Command payload* field is set to 0xfe, the *Payload* sub-field is not present, and the payload from the triggering GPD command needs to be parsed. Otherwise, a fixed payload for the Zigbee command is provided, of the *Zigbee Command payload length.*

The *Additional information block* field is formatted as illustrated in Figure 36.

Octets	1	Variable
Data Type	unsigned 8-bit integer	Sequence of unsigned 8-bit integer
Field Name	Total length of additional information	Additional information

Figure 36 – Format of the *Additional Information block* field of the GP Translation Table Update command

[76] PoC comment #19 (Zigbee document 16-02601)
[77] October PoC comment #961: https://workspace.zigbee.org/kws/groups/PRO_GP/comments/view_comment?comment_id=961
[78] Kavi comment #2107 from the GP vScVE November 2018; resolution added in 16-02607-026
[79] PoC comment #21 (Zigbee document 16-02601)

The *Total length of additional information* field indicates the total octet length of the following *Additional information block* field.

The Additional information block field is formatted as defined in sec. A.3.6.2.2.

A.3.3.4.4.1 When generated

This command is generated to configure the GPD Command Translation Table.

[80]Previous versions of this specification would not be capable of correctly processing Translation Table entries for GPD 8-bit vector press/release and GPD Compact Attribute Reporting commands, due to their inability to process the new *Additional information block* part. Before sending a GP Translation Table Update command adding translation table entries for a GPD 8-bit vector press/release or GPD Compact Attribute Reporting command, the remote node (e.g. a commissioning tool) SHOULD determine if the sink can process those Translation Table extensions (e.g. by reading the *ClusterRevision* attribute of the sink; value of 0x0002 – as defined in the current specification – indicates these Translation Table extensions are supported). If that is not the case, the remote node SHOULD NOT create translation table entries for the GPD 8-bit vector press/release or GPD Compact Attribute Reporting command.

A.3.3.4.4.2 Effect on Receipt

On receipt of this command, a sink updates its GPD Command Translation Table.

A.3.3.4.5 GP Translation Table Request command

The GP Translation Table Request command SHALL be formatted as illustrated in Figure 37.

Octets	**1**
Data Type	unsigned 8-bit integer
Field Name	Start index

Figure 37 – Format of the GP Translation Table Request command

The S*tart index* field is 8-bits in length and specifies the starting index into the GPD Command Translation Table from which to get device information. The first entry in the Translation Table has *Index* value 0.

A.3.3.4.5.1 When Generated

The GP Translation Table Request is generated to request information from the GPD Command Translation Table of remote device(s).

A.3.3.4.5.2 Effect on Receipt

Upon receipt, the sink SHALL send a GP Translation Table Response command.

[80] PoC comment #23 (Zigbee document 16-02601)

A.3.3.4.6 GP Pairing Configuration command

The GP Pairing Configuration command SHALL be formatted as illustrated in Figure 38, Figure 39 and Figure 40.

Octets	1	2	4/8	0/1	1	0/Variable	0/2
Data Type	Unsigned 8-bit integer	16-bit bitmap	Unsigned 32-bit integer/IEEE address	Unsigned 8-bit integer	8-bit enumeration	sequence of unsigned 8-bit integer	Unsigned 16-bit integer
Field Name	Actions	Options	GPD ID	Endpoint	DeviceID	GroupList	GPD Assigned Alias

Figure 38 – Format of the GP Pairing Configuration command (part 1)

1	0/1	0/4	0/16	1	0/Variable
Unsigned 8-bit integer	Unsigned 8-bit integer	Unsigned 8-bit integer	Security Key	Unsigned 8-bit integer	sequence of unsigned 8-bit integer
[81]Groupcast Radius	Security Options	GPD security frame counter	GPD security Key	Number of paired endpoints	Paired endpoints

Figure 39 – Format of the GP Pairing Configuration command (part 2)

0/1	0/2	0/2	0/1	0/Variable	0/Variable	0/Variable
8-bit bitmap	16-bit enumeration	16-bit enumeration	Unsigned 8-bit integer	Sequence of unsigned 8-bit integer	Sequence of unsigned 8-bit integer	Sequence of unsigned 8-bit integer
Application information	ManufacturerID	ModelID	Number of GPD commands	GPD CommandID list	Cluster List	Switch information

Figure 40 – Format of the GP Pairing Configuration command (part 3)

0/1	0/1	Variable	…	Variable
Unsigned 8-bit integer	Unsigned 8-bit integer	Sequence of unsigned 8-bit integer	…	Sequence of unsigned 8-bit integer
Total number of reports	Number of reports	Report descriptor M	…	Report descriptor N

Figure 41 – Format of the GP Pairing Configuration command (part 4)

A.3.3.4.6.1 Actions field

The *Actions* field is formatted as shown in Figure 42.

Bits: 0-2	3	4-7
Action	Send GP Pairing	Reserved

Figure 42 – Format of the *Actions* field of the GP Pairing Configuration command

The *Action* sub-field of the *Actions* field can take the values as defined in Table 34.

[81] CCB #2180: Resolution added in 15-02014-005

Table 34 – Values of the *Action* sub-field of the *Actions* field

Value	Description
0b000	No action.
0b001	Extend Sink Table entry.
0b010	Replace Sink Table entry.
0b011	Remove a pairing.
0b100	Remove GPD.
0b101	Application description
[82]0b110-0b111	Reserved

The *Send GP Pairing* sub-field, if set to 0b1 indicates that the receiving sink is requested to send GP Pairing command upon completing the handling of GP Pairing Configuration. If set to 0b0, it indicates that the receiving sink SHALL NOT send GP Pairing command upon completing the handling of the GP Pairing Configuration command. When the *Action* sub-field of the *Actions* field is set to 0b101, the *Send GP Pairing* sub-field of the *Actions* field SHALL be set to 0b0.

A.3.3.4.6.2 Options field

The *Options* parameter has the format as shown in Figure 43 and Figure 44.

Bits: 0..2	3..4	5	6	7
ApplicationID	Communication mode	Sequence number capabilities	RxOnCapability	FixedLocation

Figure 43 – Format of the *Options* parameter of the GP Pairing Configuration command (part 1)

8	9	10	11..15
AssignedAlias	Security use	Application information present	Reserved

Figure 44 – Format of the *Options* parameter of the GP Pairing Configuration command (part 2)

The *ApplicationID* sub-field contains the information about the application used by the GPD. *ApplicationID* = 0b000 indicates the GPD ID field has the length of 4B and contains the GPD SrcID; the *Endpoint* field is absent. *ApplicationID* = 0b010 indicates the GPD ID field has the length of 8B and contains the GPD IEEE address; the *Endpoint* field is present. All values of *ApplicationID* other than 0b000 and 0b010 are reserved in the current version of the Green Power cluster specification.

The *CommunicationMode* sub-field contains the information about the accepted tunneling mode for this GPD. It can take the values as defined in Table 27.

The *Sequence number capabilities* sub-field contains the information on the sequence number capabilities of this GPD. It takes the values as defined in sec. A.4.2.1.1.2.

The *RxOnCapability* sub-field contains the information about reception capability on this GPD.

The *FixedLocation* sub-field contains information if the location of this GPD is expected to change.

The *AssignedAlias* sub-field, if set to 0b1, indicates that the assigned alias as stored in the *GPD Assigned Alias* field SHALL be used instead of the alias derived from the GPD ID (sec. A.3.6.3.3) in case of derived groupcast or full unicast communication. If set to 0b0, the derived alias is used (sec. A.3.6.3.3) for those communication modes.

[82] Comment #705 from GP multi-sensor v0.7 letter ballot

The *Security use* sub-field, if set to 0b1, indicates that security-related fields are present.

The *Application information present* sub-field, if set to 0b1, indicates that the *Application information* field is present.

A.3.3.4.6.3 Remaining fields

All the fields *GPDID*, *Endpoint*, *DeviceID*, *GroupList*, *GPD Assigned Alias*, [83]*Groupcast Radius*, *Security Options*, *GPD security frame counter*, and *GPD security Key* are formatted as the over-the-air representation of a Sink Table entry (see sec. A.3.3.2.2).

The *Number of paired endpoints* field indicates the number of endpoints listed in the *Paired endpoints* field. If the *Number of paired endpoints* field is set to 0x00 or 0xfd, there are no paired endpoints and the *Paired endpoints* field is not present. If the *Number of paired endpoints* field is set to 0xff, all matching endpoints are to be paired and the *Paired endpoints* field is not present. If the *Number of paired endpoints* field is set to 0xfe, there paired endpoints are to be derived by the sink itself and the *Paired endpoints* field is not present.

If the *Number of paired endpoints* field has values other than 0x00, 0xfd, 0xff and 0xfe, the *Paired endpoints* field is present and contains the list of local endpoints paired to this GPD.

A.3.3.4.6.4 Application information

The fields *Application Information*, *ManufacturerID*, *ModelID*, *Number of GPD commands*, *GPD CommandID list*, *Cluster list* and *Switch information* SHALL be formatted as defined in sections A.4.2.1.1.4 -A.4.2.1.1.10.

A.3.3.4.6.5 Report description

The fields *Total number of reports*, *Number of reports*, and *Report descriptors* SHALL be formatted as defined in section A.4.2.1.6.

They SHALL only be present if the *Action* sub-field of the *Actions* field is set to 0b101; also the fields *Actions, Options, GPD ID,* in case of *ApplicationID* = 0b010 the *Endpoint* field, [84]*DeviceID*, [85][86]*Groupcast Radius*, and the *Number of paired endpoints* field SHALL be present.

The other fields: [87]*GroupList, GPD Assigned Alias,* [88]*Security Options, GPD security frame counter, GPD security Key, Application Information, ManufacturerID, ModelID, Number of GPD commands, GPD CommandID list, Cluster list* and *Switch information* SHALL be absent.

A.3.3.4.6.6 When Generated

The command is generated to configure the Sink Table of a sink, to create/update/replace/remove a pairing to a GPD and/or trigger the sending of GP Pairing command.

In the current version of the specification, a device SHALL only send GP Pairing Configuration command with the *Number of paired endpoints* field set to 0xfe, if the *CommunicationMode* is equal to Pre-Commissioned Groupcast.

A.3.3.4.6.7 [90]Effect on Receipt

On receipt of this command, the receiver is informed about the request to modify its Sink Table.

[83] CCB #2180: Resolution added in 15-02014-005
[84] CCB #2528; Resolution added in 15-02014-024
[85] CCB #2528; Resolution added in 15-02014-024
[86] CCB #2180: Resolution added in 15-02014-005
[87] CCB #2528; Resolution added in 15-02014-024
[88] CCB #2528; Resolution added in 15-02014-024
[90] Comment #703 from GP multi-sensor v0.7 letter ballot

If the *Action* sub-field of the *Actions* field is set to 0b000, only the following fields of the GP Pairing Configuration command are of importance to the receiving sink: *Send GP Pairing* sub-field, and if *Send GP Pairing* sub-field is set to 0b1, the *GPD ID* and if *ApplicationID* = 0b010, the *Endpoint* field. The other fields of the GP Pairing Configuration command: *Options*, *DeviceID*, *Pre-commissioned GroupID*, *GPD Assigned Alias*, [91]*Groupcast Radius*, *Security Options*, *GPD security frame counter*, *GPD security Key*, *Number of paired endpoints*, *Paired endpoints*, the *Application Information* fields, the *Switch information* and *Additional information block* fields, if present, are ignored.

If the *Action* sub-field of the *Actions* field is set to 0b100, only the *GPD ID* field and *Endpoint* field, if present, of the GP Pairing Configuration command is of importance to the receiving sink. The other fields of the GP Pairing Configuration command: *Options*, *DeviceID*, *GroupList*, *GPD Assigned Alias*, [92]*Groupcast Radius*, *Security Options*, *GPD security frame counter*, *GPD security Key*, *Number of paired endpoints*, *Paired endpoints*, the *Application Information* fields, the *Switch information* and *Additional information block* fields, if present, are ignored.

If the *Action* sub-field of the *Actions* field is set to a 0b011, the following fields of the received GP Pairing Configuration command are of importance: *GPD ID* field and *Endpoint* field, if present, *CommunicationMode* sub-field of the *Options* field, the *GroupList*, if present, *Number of paired endpoints*, *Paired endpoints*, if present, the *Application Information* fields, the *Switch information* and *Additional information block* fields, if present. The other fields of the received GP Pairing Configuration command: *DeviceID*, *GPD Assigned Alias*, [93]*Groupcast Radius*, *Security Options*, *GPD security frame counter*, and *GPD security Key*, if present, are ignored.

If the *Action* sub-field of the *Actions* field is set to 0b001 or 0b010, all supplied fields of the received GP Pairing Configuration command are of importance.

[94]If the *Action* sub-field of the *Actions* field is set to 0b101, the following supplied fields of the received GP Pairing Configuration command are of importance: *GPD ID* field and *Endpoint* field, if present, *Number of paired endpoints* and *Paired endpoints*, if present, thus SHALL be set to correct values upon transmission. The unconditionally present fields *DeviceID* and [95]*Groupcast Radius* SHALL be ignored upon reception and can be set to any value upon transmission. All the sub-fields of the *Options* field with the exception of the *ApplicationID* sub-field and the *Application Information present* sub-field SHALL be ignored upon reception and can be set to any value upon transmission. [96]The *Application Information present* sub-field MAY be set to 0b1; then, the *Application Information* field SHALL be present; its *GPD Application Description command follows* sub-field SHALL be set to 0b0 even if there are further GP Pairing Configuration commands with *Action*=0b101 to be sent, since the presence of further GP Pairing Configuration commands with *Action*=0b101 can be derived from the value of the fields *Total number of reports* and *Number of reports*.
[97]The sink SHALL process the individual GP Pairing Configuration commands upon reception, even if not all report descriptors have been received. [98]The sink SHALL be capable of receiving the GP Pairing Configuration command with *Action* = 0b101, i.e. carrying the *Report descriptor* information, out of order and in duplicate.

[91] CCB #2180: Resolution added in 15-02014-005
[92] CCB #2180: Resolution added in 15-02014-005
[93] CCB #2180: Resolution added in 15-02014-005
[94] October PoC comment #964: https://workspace.zigbee.org/kws/groups/PRO_GP/comments/view_comment?comment_id=964
[95] CCB #2180: Resolution added in 15-02014-005
[96] Comment #18 from GP generic switch & compact attribute reporting SVE, May 2017
[97] October PoC comment #962: https://workspace.zigbee.org/kws/groups/PRO_GP/comments/view_comment?comment_id=962
[98] Comment #777 from GP multi-sensor v0.7 letter ballot

[99]Table 35 summarizes the rules for including the various fields in the GP Pairing Configuration command.

The leftmost column after the field column recapitulates the general rules for inclusion of the particular fields, using the following notation:

- U (unconditional): the field is unconditionally present;
  - upon transmission: the field SHALL be present;
  - upon reception:
    - if the field is NOT present: the frame is malformed and SHALL be dropped without further processing.
- C (conditional):
  - upon transmission: the field MAY be present, depending on the flag settings in the *Options*, *Security Options* or *Application Information* fields;
  - upon reception:
    - if the field is NOT present while its presence is indicated by the relevant flags: the frame is malformed and SHALL be dropped without further processing.

The remaining columns indicate the rules for inclusion of the particular fields depending on the value of the *Action* sub-field of the *Actions* field, using the following notation:

- M (mandatory):
  - upon transmission: the frame SHALL be processed further;
  - upon reception:
    - if field present: its value SHALL be used;
    - if the field is NOT present: the frame is malformed and SHALL be dropped without further processing;
- O (optional):
  - upon transmission: the field MAY be present (the flag settings in the *Options*, *Security Options* or *Application Information* fields need to be set accordingly);
  - upon reception:
    - if field present (as indicated by the relevant flags): the frame SHALL be processed further;
    - if the field is NOT present while its presence is indicated by the relevant flags): the frame is malformed and SHALL be dropped without further processing;
- X (forbidden):
  - upon transmission: the field SHALL NOT be present;
  - upon reception:
    - if field NOT present: the frame SHALL be processed further;
    - if the field is present: the frame is malformed and SHALL be dropped without further processing.

In addition, the following notation is used to indicate the fields usage, if present:

- I (ignorable):
  - upon transmission: the field MAY be present (the flag settings in the *Options*, *Security Options* or *Application Information* fields need to be set accordingly);
  - upon reception: the field is ignored;

if that notation is not used for a particular field, then the value of this field, if present, SHALL be used upon reception.

[99] Dec 2016 SVE comment: https://workspace.zigbee.org/kws/groups/PRO_GP/comments/view_comment?comment_id=1029

[100]Table 35 – Presence of fields of GP Pairing Configuration commands for different values of the *Action* sub-field

Field of the GP Pairing Configuration command	General rules	Value of the *Action* sub-field of the *Actions* field of the GP Pairing Configuration command					
		0b000	0b001	0b010	0b011	0b100	0b101
Actions	U	M	M	M	M	M	M
Options	U	M	M	M	M	M	M
GPD ID	U	M	M	M	M	M	M
Endpoint	C	O	O	O	O	O	O
DeviceID	U	M : I	M	M	M : I	M : I	M :[101] I
GroupList	C	O : I	O	O	O	O : I	X
GPD Assigned Alias	C	O : I	O	O	O : I	O : I	X
[102]Groupcast Radius	U	M : I	M	M	M : I	M : I	M: [103] I
Security Options	C	O : I	O	O	O : I	O : I	X
GPD security frame counter	C	O: I	O	O	O: I	O: I	X
GPD security key	C	O: I	O	O	O : I	O : I	X
Number of paired endpoints	U	M: I	M	M	M	O : I	[104]M
Paired endpoints	C	O : I	O	O	O	O : I	[105]O
Application information	C	O : I	O	O	O	O : I	[106]O
ManufacturerID	C	O : I	O	O	O	O : I	X
ModelID	C	O : I	O	O	O	O : I	X
Number of GPD commands	C	O : I	O	O	O	O : I	X
GPD CommandID list	C	O : I	O	O	O	O : I	X
Cluster List	C	O : I	O	O	O	O : I	X
Switch information	C	O : I	O	O	O	O : I	X
Total number of reports	C	O : I	O	O	O	O : I	M
Number of reports	C	O : I	O	O	O	O : I	M
Report descriptor(s)	C	O : I	O	O	O	O : I	M

[100] Dec 2016 SVE comment: https://workspace.zigbee.org/kws/groups/PRO_GP/comments/view_comment?comment_id=1029
[101] October PoC comment #964: https://workspace.zigbee.org/kws/groups/PRO_GP/comments/view_comment?comment_id=964
[102] CCB #2180: Resolution added in 15-02014-005
[103] October PoC comment #964: https://workspace.zigbee.org/kws/groups/PRO_GP/comments/view_comment?comment_id=964
[104] October PoC comment #964: https://workspace.zigbee.org/kws/groups/PRO_GP/comments/view_comment?comment_id=964
[105] October PoC comment #964: https://workspace.zigbee.org/kws/groups/PRO_GP/comments/view_comment?comment_id=964
[106] Comment #18 from GP generic switch & compact attribute reporting SVE, May 2017

zigbee alliance

A.3.3.4.7 GP Sink Table Request command

The payload of the GP Sink Table Request command SHALL be formatted as illustrated in Figure 45.

Octets	1	0/4/8	0/1	0/1
Data Type	8-bit bitmap	unsigned 32-bit integer/IEEE address	Unsigned 8-bit integer	unsigned 8-bit integer
Field Name	Options	GPD ID	Endpoint	Index

Figure 45 – Format of the GP Sink Table Request command

The *Options* field of the GP Sink Table Request command is formatted as shown in Figure 46.

Bits: 0..2	3..4	5..7
ApplicationID	Request type	Reserved

Figure 46 – Format of the Options field of the GP Sink Table Request command

The *ApplicationID* sub-field contains the information about the application used by the GPD. *ApplicationID* = 0b000 indicates the *GPD ID* field, if present as indicated by the *Request type* sub-field of the *Options* field, has the length of 4B and contains the GPD SrcID; the *Endpoint* field is absent. *ApplicationID* = 0b010 indicates the *GPD ID*, if present as indicated by the *Request type* sub-field of the *Options* field, field has the length of 8B and contains the GPD IEEE address; the *Endpoint* field is present if the IEEE address is present. All values of *ApplicationID* other than 0b000 and 0b010 are reserved in the current version of the Green Power cluster specification.

The *Request type* sub-field specifies how table entries are requested. It SHALL take one of the non-reserved the values defined in Table 36.

Table 36 – Values of the *Request type* sub-field of the *Options* field of the GP Sink Table Request command

Value	Description
0b00	Request table entries by GPD ID
0b01	Request table entries by Index
0b10 – 0b11	Reserved

If set to 0b00, it indicates that the *GPD ID* field, and *Endpoint* field, if *ApplicationID* = 0b010, is present and carries the GPD ID for which the Sink Table entry is requested; the *Index* field is absent.

If set to 0b01, it indicates that the *Index* field is present and carries the starting index for the Sink Table entry request; the *GPD ID* field and the *Endpoint* field are absent.

The *GPD ID* field carries the value of the *GPD ID*, either GPD SrcID or GPD IEEE address, depending on the value of the *ApplicationID*, for which the Sink Table entry is requested.

The *Endpoint* field carries the value of the GPD endpoint for which the Sink Table entry is requested.

The *Index* field carries the index value of the Sink Table entry is requested. The index enumeration includes only non-empty Sink Table entries. It starts with 0x00; 0xff indicates unspecified.

A.3.3.4.7.1 When generated

The GP Sink Table Request command is generated to read out selected Sink Table entry(s), by index or by GPD ID (and Endpoint if *ApplicationID* = 0b010).

If the sender of the command wishes to avoid receiving many responses, esp. from the nodes not supporting this functionality, it SHALL set the *Disable default response* sub-field of the *Frame Control* field of the ZCL header of the GP Sink Table Request command, as specified in sec. 2.3.1.1.4 of [3].

A.3.3.4.7.2 Effect on receipt

On receipt of this command, the device is informed about a request for selected Sink Table entries.

A.3.3.4.8 GP Sink Commissioning Mode command

The payload of the GP Sink Commissioning Mode command SHALL be formatted as illustrated in Figure 47.

Octets	1	2	2	1
Data Type	8-bit bitmap	16-bit unsigned integer	16-bit unsigned integer	8-bit unsigned integer
Field Name	Options	GPM address for security	GPM address for pairing	Sink Endpoint

Figure 47 – Format of the GP Sink Commissioning Mode command

The *Options* field of the GP Sink Commissioning Mode command is formatted as shown in Figure 48.

Bits: 0	1	2	3	4..7
Action	Involve GPM in security	Involve GPM in pairing	Involve proxies	Reserved

Figure 48 – Format of the Options field of the GP Sink Commissioning Mode command

The *Action* field indicates the operation to be performed by the sink on reception. If set to 0b1, the sink is requested to enter commissioning mode. If set to 0b0, the sink is requested to exit commissioning mode.

The *Involve GPM in security* sub-field indicates how the security check during the commissioning action being enabled is to be performed. If the *Involve GPM in security* sub-field is set to 0b0, the receiving sink is requested to perform security matching itself; the *GPM address for security* is ignored. If the *Action* field is set to 0b0, the *Involve GPM in security* sub-field is ignored. In the current version of the specification, the *Involve GPM in security* sub-field SHALL be set to 0b0.

The *Involve GPM in pairing* sub-field indicates how the application functionality matching during the commissioning action being enabled is to be performed. If the *Involve GPM in pairing* sub-field is set to 0b0, the receiving sink is requested to perform application functionality matching (see sec. A.3.6.2.1) itself; the *GPM address for pairing* is ignored. If the *Action* field is set to 0b0, the *Involve GPM in pairing* sub-field is ignored. In the current version of the specification, the *Involve GPM in pairing* sub-field SHALL be set to 0b0.

The *Involve proxies* sub-field indicates if proxies SHALL be involved in the commissioning action being enabled. If set to 0b1, the sink is requested, upon entering or exiting the commissioning mode, as specified by the *Action* sub-field of the *Options* field of the received GP Sink Commissioning Mode command, to send the GP Proxy Commissioning Mode command with the same *Action* sub-field value.

The *GPM address for security* field SHALL be set to 0xffff in the current version of the specification.

The *GPM address for pairing* field SHALL be set to 0xffff in the current version of the specification.

The *Sink Endpoint* field indicates for which application endpoint the Green Power commissioning is requested to be enabled. The value of 0xff indicates all active endpoints.

A.3.3.4.8.1 When generated

The GP Sink Commissioning Mode command is generated by a remote device, e.g. a Commissioning Tool, to request a sink to perform a commissioning action in a particular way.

A.3.3.4.8.2 Effect on receipt

On receipt of this command, the device is informed about a request for a particular commissioning action.

If the sink does not implement the endpoint indicated by the *Sink Endpoint* field, it SHALL NOT enter the commissioning mode. It SHALL then send a ZCL default response with the Status NOT_FOUND (for the values of the Status codes see [3]).

If the sink not supporting Multi-hop commissioning receives GP Sink Commissioning Mode with *InvolveProxies* = 0b1, it SHALL enter the commissioning mode it supports, incl. proximity commissioning; it SHALL NOT send the GP Proxy Commissioning Mode command.

If the sink not supporting proximity commissioning receives GP Sink Commissioning Mode with *InvolveProxies* = 0b0, it SHALL enter the commissioning mode it supports, incl. Multi-hop commissioning; it SHALL NOT send the GP Proxy Commissioning Mode command.

If the fields *GPM address for security* or *GPM address for pairing* carry value other than 0xffff or any of *Involve GPM in security* or *Involve GPM in pairing* sub-fields of the *Options* field is set, a sink implemented according to the current specification it SHALL NOT enter the commissioning mode. It SHALL then send a ZCL default response with the *Status* INVALID_VALUE [107]or INVALID_FIELD; it is recommended that INVALID_FIELD value is returned (see [3]).

If the sender of the command wishes to avoid receiving many responses, esp. from the nodes not supporting this functionality, it SHALL set the *Disable default response* sub-field of the *Frame Control* field of the ZCL header of the GP Sink Commissioning Mode command, as specified in sec. 2.3.1.1.4 of [3].

After entering the commissioning mode upon reception of GP Sink Commissioning Mode command with *Action* = Enter, the sink SHALL exit the commissioning mode either by the default exit condition, as specified in the *gpsCommissioningExitMode* attribute, or upon reception of GP Sink Commissioning Mode command with *Action* = Exit.

[107] CCB #2337; Resolution added in 15-02014-009

A.3.3.5 Commands generated

Whether the support of particular command is mandatory or optional is dependent on the GP infrastructure device type and the functionality it supports, and specified in Table 23.

Table 37 – Green Power cluster: server side: commands generated

Command Value	Command Name	Command Description	Link
0x00	GP Notification Response	From sink to a proxy to acknowledge GP Notification received in full unicast mode.	A.3.3.5.1
0x01	GP Pairing	From sink to the entire network to (de)register for tunneling service, or for removing GPD from the network	A.3.3.5.2
0x02	GP Proxy Commissioning Mode	From sink to proxies in the whole network to indicate commissioning mode	A.3.3.5.3
0x03-0x05	Reserved		
0x06	GP Response	From sink to selected proxies, to provide data to be transmitted to Rx-capable GPD	A.3.3.5.4
0x07	Reserved		
0x08	GP Translation Table Response	To provide GPD Command Translation Table content	A.3.3.5.5
0x09	Reserved		
0x0a	GP Sink Table Response	To send selected Sink Table entries	A.3.3.5.6
0x0b	GP Proxy Table Request	To requested selected Proxy Table entries	A.3.4.3.1
0x0c – 0xff	Reserved		

A.3.3.5.1 GP Notification Response command

The payload of the GP Notification Response command SHALL be formatted as illustrated in Figure 49.

Octets	1	4/8	0/1	4
Data Type	8-bit bitmap	unsigned 32-bit integer/IEEE address	Unsigned 8-bit integer	Unsigned 32-bit integer
Field Name	Options	GPD ID	Endpoint	GPD security frame counter

Figure 49 – Format of the GP Notification Response command

The *Options* field SHALL be formatted as shown in Figure 50.

Bits: 0..2	3	4	5..7
ApplicationID	FirstToForward	NoPairing	Reserved

Figure 50 – Format of the Options field of the GP Notification Response command

The *ApplicationID* sub-field contains the information about the application used by the GPD. *ApplicationID* = 0b000 indicates the GPD ID field has the length of 4B and contains the GPD SrcID; the *Endpoint* field is absent. *ApplicationID* = 0b010 indicates the GPD ID field has the length of 8B and contains the GPD IEEE address; the *Endpoint* field is present. All values of *ApplicationID* other than 0b000 and 0b010 are reserved in the current version of the Green Power cluster specification.

The *FirstToForward* sub-field indicates if the GP Notification from this proxy was the first for this GPDF. If set to 0b1, the proxy's GP Notification reached the sink as first for this GPD and Frame Counter value. If set to 0b0, it was a duplicate.

The *NoPairing* sub-field, when set to 0b1, indicates that the sink has no pairing with this GPD ID (and *Endpoint*, if *ApplicationID* = 0b010).

The *GPD security frame counter* is copied from the GP Notification.

A.3.3.5.1.1 When generated

This command is generated when the sink acknowledges the reception of full unicast GP Notification command.

The GP Notification Response command is sent in unicast to the originating proxy.

A.3.3.5.1.2 Effect on Receipt

On receipt of the GP Notification Response command, a proxy is informed about sink having received a full unicast GP Notification.

A.3.3.5.2 GP Pairing command

The payload of the GP Pairing command SHALL be formatted as illustrated in Figure 51 and Figure 52.

Octets	**3**	**4/8**	**0/1**	**0/8**	**0/2**	**0/2**
Data Type	24-bit bitmap	unsigned 32-bit integer/IEEE address	Unsigned 8-bit integer	IEEE address	unsigned 16-bit integer	unsigned 16-bit integer
Field Name	Options	GPD ID	Endpoint	Sink IEEE address	Sink NWK address	Sink GroupID

Figure 51 – Format of the GP Pairing command (part 1)

0/1	**0/4**	**0/16**	**0/2**	**0/1**
8-bit enumeration	unsigned 32-bit integer	Security key	unsigned 16-bit integer	Unsigned 8-bit integer
DeviceID	GPD security Frame Counter	GPD key	Assigned alias	[108]Groupcast Radius

Figure 52 – Format of the GP Pairing command (part 2)

The *Options* field of the GP Pairing command SHALL be formatted as illustrated in Figure 53 and Figure 54.

Bits: 0..2	**3**	**4**	**5..6**	**7**	**8**	**9..10**
ApplicationID	AddSink	RemoveGPD	CommunicationMode	GPD Fixed	GPD MAC sequence number capabilities	SecurityLevel

Figure 53 – Format of the *Options* field of the GP Pairing command (part 1)

11..13	**14**	**15**	**16**	**17**	**18..23**
SecurityKeyType	GPD security Frame Counter present	GPDsecurityKeyPresent	Assigned Alias present	[109]Groupcast Radius present	Reserved

Figure 54 – Format of the *Options* field of the GP Pairing command (part 2)

[108] CCB #2180: Resolution added in 15-02014-005
[109] CCB #2180: Resolution added in 15-02014-005

The *ApplicationID* sub-field contains the information about the application used by the GPD. *ApplicationID* = 0b000 indicates the GPD ID field has the length of 4B and contains the GPD SrcID; the *Endpoint* field is absent. *ApplicationID* = 0b010 indicates the GPD ID field has the length of 8B and contains the GPD IEEE address; the *Endpoint* field is present. All values of *ApplicationID* other than 0b000 and 0b010 are reserved in the current version of the Green Power cluster specification.

The *AddSink* sub-field of the *Options* field indicates, whether the GP sink wishes to add or remove a pairing for the GPD identified by the *GPD ID*. If set to 0b1 the pairing is being added. If set to 0b0 the pairing is being removed; then, the following fields are not present: *DeviceID*, *GPD security Frame Counter*, *GPD key*, *AssignedAlias*, and [110]*Groupcast Radius*.

The *RemoveGPD* sub-field of the *Options* field, if set to 0b1, indicates that the GPD identified by the *GPD ID* is being removed from the network. Then, none of the optional fields is present.

The *CommunicationMode* sub-field defines the communication mode requested by the sink, and can take values as defined in Table 27.

The *GPDfixed* sub-field and *GPD MAC sequence number capabilities* sub-field is copied from the corresponding *FixedLocation* and *Sequence number capabilities* sub-fields of the *Options* parameter of the Sink Table for this GPD.

The *SecurityLevel* and *SecurityKeyType* SHALL carry the values of the corresponding parameters in Sink Table entry for this GPD.

The sub-fields *GPDsecurityFrameCounterPresent* and *GPDsecurityKeyPresent*, if set to 0b1, indicate the presence of the fields *GPDsecurityFrameCounter* and *GPDsecurityKey*, respectively, which then carry the corresponding values from the Sink Table for this GPD. When the sub-fields *GPDsecurityFrameCounterPresent* and *GPDsecurityKeyPresent* are set to 0b0, the fields *GPDsecurityFrameCounter* and *GPDsecurityKey*, respectively, are not present.

If the *SecurityLevel* is 0b00 and the *GPD MAC sequence number capabilities* sub-field is set to 0b0, the *GPDsecurityFrameCounter* field SHALL NOT be present, the *GPDsecurityFrameCounterPresent* sub-field of the *Options* field SHALL be set to 0b0.

The *GPDsecurityFrameCounter* field SHALL be present [111]and the *GPDsecurityFrameCounterPresent* sub-field of the *Options* field SHALL be set to 0b1 whenever the *AddSink* sub-field of the *Options* field is set to 0b1 [112]and one of the following cases applies:

- if the *SecurityLevel* sub-field is set to 0b10 or 0b11 or;
- if the *SecurityLevel* is 0b00 and the *GPD MAC sequence number capabilities* sub-field is set to 0b1.

[113]The *GPDsecurityFrameCounter* field then carries the current value of the *GPD security frame counter* field from the Sink Table entry corresponding to the *GPD ID*.

If the *SecurityLevel* is 0b00 and the *GPD MAC sequence number capabilities* sub-field is set to 0b0, the *GPDsecurityFrameCounter* SHALL [114]NOT be present, the *GPDsecurityFrameCounterPresent* sub-field of the *Options* field SHALL be set to 0b0.

[110] CCB #2180: Resolution added in 15-02014-005
[111] CCB #2272; Resolution added in 15-02014-006;
[112] CCB #2344; Resolution added in 15-02014-011;
[113] CCB #2344; Resolution added in 15-02014-011;
[114] CCB #2344; Resolution added in 15-02014-011;

The *AssignedAlias present* sub-field, if set to 0b1, indicates that the *AssignedAlias* field is present and carries the Alias value to be used for this GPD instead of the derived alias.

The [115]*Groupcast Radius present* sub-field, if set to 0b1, indicates that the [116]*Groupcast Radius* field is present and carries the [117]*Groupcast Radius* value to be used as value of the radius in the groupcast forwarding of the GPDF packet. If the [118]*Groupcast Radius* field is not present, and a new Proxy Table entry is to be created, the default value of 0x00 SHALL be used. The value 0x00 indicates unspecified, i.e. twice the value of the nwkMaxDepth attribute of the NIB, as specified by [1].

The *GPD ID* field carries the value of the GPD identifier, either GPD SrcID or GPD IEEE address of the GPD for which the pairing is being managed.

The *Endpoint* field carries the value of the GPD endpoint for which the pairing is being managed.

The presence of the addressing fields (*SinkIEEEaddress*, *SinkNWKaddress*, and *SinkGroupID*) is indicated by the sub-fields *RemoveGPD* and the *CommunicationMode* of the *Options* field, as shown in Table 38 below. Any of the fields can only be present, if the *RemoveGPD* sub-field is set to 0b0. The fields *SinkIEEEaddress* and *SinkNWKaddress* are only present if full or lightweight unicast communication mode is requested. The *SinkGroupID* field is only present, if one of the groupcast communication modes is requested.

Table 38 – Presence of the addressing fields in the GP Pairing command

RemoveGPD value	CommunicationMode value	SinkIEEEaddress and SinkNWKaddress present	SinkGroupID present
0b1	Any	X	X
0b0	0b00 or 0b11	M	X
0b0	0b01	X	M
0b0	0b10	X	M

The *SinkIEEEaddress* and *SinkNWKaddress*, if present, carry the IEEE address and the NWK address, respectively, of the sink originating the GP Pairing command.

The *SinkGroupID* field, if present, carries the GroupID the sink originating the GP Pairing command is member of.

[119]If the sender of the command wishes to avoid receiving many responses, especially from the nodes not supporting this functionality, it SHALL set the *Disable default response* sub-field of the *Frame Control* field of the ZCL header of the GP Pairing command, as specified in sec. 2.3.1.1.4 of [3].

[115] CCB #2180: Resolution added in 15-02014-005
[116] CCB #2180: Resolution added in 15-02014-005
[117] CCB #2180: Resolution added in 15-02014-005
[118] CCB #2180: Resolution added in 15-02014-005
[119] CCB #2394; Resolution added in 15-02014-010

A.3.3.5.2.1 When generated

The GP Pairing command is generated by the sink to manage pairing information.

The GP Pairing command is typically sent using network-wide broadcast.

If the *CommunicationMode* sub-field is set to 0b11, GP Pairing command MAY be sent in unicast to the selected proxy.

A.3.3.5.2.2 Effect on Receipt

On receipt of this command, a device is informed about pairing update (creation or deletion).

A.3.3.5.3 GP Proxy Commissioning Mode command

The payload of the GP Proxy Commissioning Mode command SHALL be formatted as shown in Figure 55.

Octets	**1**	**0/2**	**0/1**
Data Type	8-bit bitmap	Unsigned 16-bit integer	Unsigned 8-bit integer
Field Name	Options	CommissioningWindow	Channel

Figure 55 – Format of the GP Proxy Commissioning Mode command

The *Options* field SHALL be formatted as shown in Figure 56.

Bits: 0	**1**	**2-3**	**4**	**5**	**6-7**
Action	[120]Commissioning-ingWindow present	Exit mode	Channel present	Unicast communication	Reserved

Figure 56 – Format of the Options field of the GP Proxy Commissioning Mode command

The *Action* sub-field, if set to 0b1, indicates a request to enter commissioning mode. If set to 0b0, it indicates a request to exit commissioning mode.

[121]The *CommissioningWindow present* sub-field, if set to 0b1, indicates that the *CommissioningWindow* field is present. If set to 0b0, the *CommissioningWindow* field is absent.

The *Exit mode* sub-field SHALL be formatted as shown in Figure 57. When the *Action* sub-field is set to 0b1, the *Exit mode* sub-field carries the value of the *gpsCommissioningExitMode* attribute (see A.3.3.2.5). When the *Action* sub-field is set to 0b0, the value of the *Exit mode* sub-field is ignored.

	Bits: 0	**1**
	On first Pairing success	On GP Proxy Commissioning Mode (exit)

Figure 57 – Format of the Exit mode sub-field of the Options field of the GP Proxy Commissioning Mode command

The *Channel present* sub-field of the *Options* field, if set to 0b0, indicates that the devices SHOULD go to (or stay on) the operational channel. If set to 0b1, it indicates that the *Channel* field is present, which carries the identifier of the channel the devices SHOULD switch to on reception (e.g. 0x0b for channel 11). The value 0xff indicates unspecified.

In the current version of the GP specification, the *Channel present* sub-field SHALL always be set to 0b0 and the *Channel* field SHALL NOT be present.

[120] CCB #2353; Resolution added in 15-02014-011
[121] CCB #2353; Resolution added in 15-02014-011

The *Unicast communication* sub-field of the *Options* field, if set to 0b0, indicates that the receiving proxies SHALL send the GP Commissioning Notification commands in broadcast. If set to 0b1, it indicates that the receiving proxies SHALL send the GP Commissioning Notification commands in unicast to the originator of the GP Proxy Commissioning Mode command. When the *Action* sub-field is set to 0b0, the value of the *Unicast communication* sub-field is ignored.

[122]The *CommissioningWindow* field SHALL be present, if the *CommissioningWindow present* sub-field of the *Options* field is set to 0b1. It carries the value of *gpsCommissioningWindow* attribute (see A.3.3.2.5), which overrides - for this particular commissioning operation - the default *gppCommissioningWindow* value (see A.3.6.3.2) of the receiving proxy.

[123]If the sender of the command wishes to avoid receiving many responses, especially from the nodes not supporting this functionality, it SHALL set the *Disable default response* sub-field of the *Frame Control* field of the ZCL header of the GP Proxy Commissioning Mode command, as specified in sec. 2.3.1.1.4 of [3].

A.3.3.5.3.1 When generated

This command is generated when the sink wishes to instruct the proxies to enter/exit commissioning mode. [124]The GP Proxy Commissioning Mode command is typically sent using network-wide broadcast.

A.3.3.5.3.2 Effect on Receipt

On receipt of this command, a device is instructed about requested commissioning actions.

[122] CCB #2353; Resolution added in 15-02014-011
[123] CCB #2394; Resolution added in 15-02014-010
[124] CCB #2122: Resolution added in 15-02014-002

A.3.3.5.4 GP Response command

The payload of the GP Response command SHALL be formatted as illustrated in Figure 58.

Octets	1	2	1	4/8	0/1	1	Variable
Data Type	Unsigned 8-bit integer	Unsigned 16-bit integer	8-bit bitmap	Unsigned 32-bit integer/IEEE address	Unsigned 8-bit integer	Unsigned 8-bit integer	Octet string
Field Name	Options	TempMaster short address	TempMaster Tx channel	GPD ID	Endpoint	GPD Comman-dID	GPD Command payload

Figure 58 – Format of the GP Response command

The *Options* SHALL be formatted as shown in Figure 60.

Bits: 0..2	3	4..7
ApplicationID	Transmit on end-point match	Reserved

Figure 59 – Format of the Options field of the GP Response command

The *ApplicationID* sub-field contains the information about the application used by the GPD. *ApplicationID* = 0b000 indicates the GPD ID field has the length of 4B and contains the GPD SrcID; the *Endpoint* field is absent. *ApplicationID* = 0b010 indicates the GPD ID field has the length of 8B and contains the GPD IEEE address; the *Endpoint* field is present. All values of *ApplicationID* other than 0b000 and 0b010 are reserved in the current version of the Green Power cluster specification.
The *Transmit on endpoint match* sub-field indicates how the sender of the GP Response command intends for the GPD command to be transmitted by the TempMaster. If *ApplicationID* = 0b010, and the *Transmit on endpoint match* = 0b1, the TempMaster is requested to deliver the frame when the GPD IEEE address and the *Endpoint* field of the received GPDF with *RxAfterTx* match exactly the values supplied in the GP Response. If *ApplicationID* = 0b010, and the *Transmit on endpoint match* = 0b0, the TempMaster is requested to deliver the frame when the GPD IEEE address of the received GPDF with *RxAfterTx* matches the values supplied in the GP Response; the value of the *Endpoint* field is ignored. If the *ApplicationID* = 0b000, this sub-field is ignored.

The *TempMaster short address* field indicates the address of the proxy which will transmit the response GPDF to the GPD.

The *TempMaster Tx Channel* field indicates the channel the Response GPDF will be sent on. It SHALL be formatted as shown in Figure 60.

Bits: 0-3	4-7
Transmit channel	Reserved

Figure 60 – Format of the TempMaster Tx Channel field of the GP Response command

The *Transmit channel* sub-field of the *TempMaster Tx Channel* field can take the following values: 0b0000: channel 11, 0b0001: channel 12, … , 0b1111: channel 26.

The *GPD ID* field carries the identifier of the GPD for which the GPDF frame is intended. If the GPD command is to be sent with the Maintenance *Frame Type*, the *ApplicationID* sub-field of the *Options* field SHALL be set to 0b000 and the *GPD ID* SHALL carry the value 0x00000000.

The fields *GPD CommandID* and *GPD Command payload* carry the input for the GPDF.

The *GPD Command Payload* field is an octet string. The first octet contains the payload length; the following octets – the value for the GPDF *Command payload* field. The value of 0xff indicates unspecified/no payload; 0x00 indicates no payload.

A.3.3.5.4.1 When generated

This command is generated when sink requests to send any information to a specific GPD with Rx capability.

A.3.3.5.4.2 Effect on Receipt

See A.3.5.2.1.

A.3.3.5.5 GP Translation Table Response command

The GP Translation Table Response command SHALL be formatted as illustrated in Figure 61.

Octets	1	1	1	1	1	Variable
Data Type	8-bit enumeration	Unsigned 8-bit integer	unsigned 8-bit integer	unsigned 8-bit integer	unsigned 8-bit integer	N*Variable
Field Name	Status	Options	Total number of entries	Start index	Entries count	TranslationTableList

Figure 61 – Format of the GP Translation Table Response command

[125]The *Status* field can take the value of SUCCESS (for the values of the Status codes see [3]).

The *Options* SHALL be formatted as shown in Figure 60.

Bits: 0..2	3	4..7
ApplicationID	Additional information block present	Reserved

Figure 62 – Format of the Options field of the GP Translation Table Response command

The *ApplicationID* sub-field contains the information about the application used by the GPD. *ApplicationID* = 0b000 indicates the GPD ID field of each Translation Table entry in the *TranslationTableList* field has the length of 4B and contains the GPD SrcID; the *GPD Endpoint* field is absent. *ApplicationID* = 0b010 indicates the GPD ID field has the length of 8B and contains the GPD IEEE address; the *GPD Endpoint* field is present. All values of *ApplicationID* other than 0b000 and 0b010 are reserved in the current version of the Green Power cluster specification.

The *Additional information block present* sub-field, if set to 0b1, indicates that the *Additional information block* field is present; if set to 0b0, it indicates that the *Additional information block* field is absent.

The *Total number of entries* field specifies the number of entries in the GPD Command Translation Table (see Table 48) of this sink.

The S*tart index* field specifies the starting index into the GPD Command Translation Table of this sink from which the information is included. This value of this field SHALL be equal to the value of the *start index* field GP Translation Table Request command. The first entry in the Translation Table has *Index* value 0.

The *Entries count* field specifies the number *N* of entries in the *TranslationTableList* field.

Each entry in the *TranslationTableList* is formatted as shown in Figure 63 and Figure 64. The entries in the *TranslationTableList* field are ordered by *Index* field value, with the lowest entry being sent first.

[125] PoC comment #11 (Zigbee document 16-02601)

Octets	4/8	0/1	1	1	2	2
Data Type	unsigned 32-bit integer/IEEE address	Unsigned 8-bit integer	unsigned 8-bit integer	unsigned 8-bit integer	unsigned 16-bit integer	unsigned 16-bit integer
Field Name	GPD ID	GPD Endpoint	GPD Command ID	EndPoint	Profile	Cluster

Figure 63 – Format of the entry of the TranslationTableList field of the GP Translation Table Response command (part 1)

1	1	0/Variable	0/Variable
unsigned 8-bit integer	unsigned 8-bit integer	Sequence of unsigned 8-bit integer	Sequence of unsigned 8-bit integer
Zigbee Command ID	Zigbee Command payload length	Zigbee Command payload	Additional information block

Figure 64 – Format of the entry of the TranslationTableList field of the GP Translation Table Response command (part 2)

If the *Endpoint* field is set to 0xff, the translation applies to all matching endpoints. If the *Endpoint* field is set to 0xfd, there are no endpoints to which this translation applies.

The *Zigbee Command payload length* field indicates the length of the *Zigbee Command payload* field. If the *Zigbee Command payload length* field is set to 0x00, there is no payload.

[126]The *Additional information block* field is formatted as defined in Figure 82.

A.3.3.5.5.1 When Generated

The GP Translation Table Response command is generated by a sink on reception of a GP Translation Table Request command.

[127]When the GPD Command Translation Table is empty or when the *Start Index* field value from the triggering Translation Table Request command exceeds the total number of entries in GPD Command Translation Table is empty, the sink implemented according to the current version of the specification SHALL return GP Translation Table Response command with the value NOT_FOUND in the *Status* field (see [3]) and the correct value in the *Total number of entries* field (0x00 in case of empty GPD Command Translation Table); the fields *Options*[128] and *Entries count* SHALL be set to 0x00; [129]the *Start index* field SHALL be set to either to 0x00 or to the value of the *Start index* field from the triggering GP Translation Table Request command; the *TranslationTableList* field SHALL NOT be included.

[130]Note: Sinks implemented according to the previous versions of this specification return, when the GPD Command Translation Table is empty, the GP Translation Table Response command with the value SUCCESS in the *Status* field (see [3]) and 0x00 in the *Total number of entries* field.

[131]If the Translation Table functionality is not supported, the sink returns ZCL Default response command, with the status UNSUP_CLUSTER_COMMAND (see [3]).

[126] Comment #776 from GP multi-sensor v0.7 letter ballot
[127] PoC comment #12, #15 (Zigbee document 16-02601)
[128] Kavi comment #2106 from the GP vScVE November 2018; resolution added in 16-02607-026
[129] Kavi comment #2106 from the GP vScVE November 2018; resolution added in 16-02607-026
[130] PoC comment #12 (Zigbee document 16-02601)
[131] PoC comment #11 (Zigbee document 16-02601)

[132]If not even a single Translation Table entry fits in the GP Translation Table Response command, the sink SHALL return GP Translation Table Response command with the value INSUFFICIENT_SPACE in the *Status* field (see [3]) and the correct value in the *Total number of entries* field; the fields *Options*, *Start index* and *Entries count* SHALL be set to 0x00; the *TranslationTableList* field SHALL NOT be included.

A.3.3.5.5.2 Effect on Receipt

The receiving device gets information on the GPD Command Translation Table of the sink that sent the command.

A.3.3.5.6 GP Sink Table Response command

The GP Sink Table Response command SHALL be formatted as illustrated in Figure 65.

Octets	**1**	**1**	**1**	**1**	**0/Variable**	**…**	**0/Variable**
Data Type	8-bit enumeration	Unsigned 8-bit integer	unsigned 8-bit integer	unsigned 8-bit integer	Octet string	…	Octet string
Field Name	Status	Total number of non-empty Sink Table entries	Start index	Entries count	Sink Table entry	,,,	Sink Table entry

Figure 65 – Format of the GP Sink Table Response command

The *Status* field can take the values of SUCCESS or NOT_FOUND (for the values of the Status codes see [3]).

The *Total number of non-empty Sink Table entries* field specifies the total number of non-empty Sink Table entries currently available on the responding device. Value of 0x00 indicates the Sink Table is empty. Value of 0xff indicates Sink Table is not implemented.

The *Start index* field specified the table position of the first of the Sink Table entry included. The first non-empty entry in the Sink Table has *Index* value 0.

The *Entries count* field specifies the number of *Sink Table entry* fields included in the current message.

Each *Sink Table entry* field contains a complete Sink Table entry, formatted as specified in sec. A.3.3.2.2.1. The entries are ordered by *Index* field value, with the lowest entry being sent first.

A.3.3.5.6.1 When generated

Upon reception of the GP Sink Table Request command, the device SHALL check if it implements a Sink Table.

If not, it SHALL generate a ZCL Default Response command, with the *Status code* field carrying UNSUP_CLUSTER_COMMAND, subject to the rules as specified in sec. 2.4.12 of [3].

If the device implements the Sink Table, it SHALL prepare a GP Sink Table Response.

If its Sink Table is empty, and the triggering GP Sink Table Request was received in unicast, then the GP Sink Table Response SHALL be sent with *Status* [133]NOT_FOUND, *Total number of non-empty Sink Table entries* carrying 0x00, *Start index* carrying 0xFF (in case of request by GPD ID) or the *Index* value from the triggering GP Sink Table Request (in case of request by index), *Entries count* field set to 0x00, and any *Sink Table entry* fields absent.

[132] PoC comment #14, #15 (Zigbee document 16-02601)
[133] CCB #2171; Resolution added in 15-02014-005

If the triggering GP Sink Table Request command contained an *Index* field, the device SHALL check if it has at least Index+1 non-empty Sink Table entries. If not, the device SHALL create a GP Sink Table Response with *Status* NOT_FOUND, *Total number of non-empty Sink Table entries* carrying the total number of non-empty Sink Table entries on this device, *Start index* carrying the *Index* value from the triggering GP Sink Table Request, *Entries count* field set to 0x00 and any *Sink Table entry* fields absent. If yes, the device SHALL create a GP Sink Table Response with *Status* SUCCESS, *Total number of non-empty Sink Table entries* carrying the total number of non-empty Sink Table entries on this device, *Start index* carrying the *Index* value from the triggering GP Sink Table Request, *Entries count* field set to the number of complete non-empty Sink Table entries, which are included in this response, followed by those *Sink Table entry* fields themselves, formatted as specified in sec. A.3.3.2.2.1.

Note: the device SHALL only include complete Sink Table entries; if an entry does not fit completely into the frame, it SHALL NOT be included in this Response.
Note 2: If there are empty Sink Table entries between non-empty Sink Table entries, they SHALL NOT be included in the response.

If the triggering GP Sink Table Request command contained a *GPD ID* field, the device SHALL check if it has a Sink Table entry for this GPD ID (and Endpoint, if *ApplicationID* = 0b010). If yes, the device SHALL create a GP Sink Table Response with *Status* SUCCESS, *Total number of non-empty Sink Table entries* carrying the total number of non-empty Sink Table entries on this device, *Start index* set to 0xff, *Entries count* field set to 0x01, and one *Sink Table entry* field for the requested GPD ID (and Endpoint, if *ApplicationID* = 0b010), formatted as specified in sec. A.3.3.2.2.1, present.

If the entry requested by GPD ID (and Endpoint, if *ApplicationID* = 0b010) cannot be found, and the triggering GP Sink Table Request was received in unicast, then the GP Sink Table Response SHALL be sent with *Status* NOT_FOUND, *Total number of non-empty Sink Table entries* carrying the total number of non-empty Sink Table entries on this device, *Start index* carrying 0xFF, *Entries count* field set to 0x00, and any *Sink Table entry* fields absent. If the triggering GP Sink Table Request was received in groupcast or broadcast, then the GP Sink Table Response SHOULD be skipped.

A.3.3.5.6.2 Effect on receipt

On receipt of this command, the remote device is informed about selected Sink Table entries on the sending device.

A.3.4 Client

A.3.4.1 Dependencies

None.

A.3.4.2 Attributes

The client side of the Green Power cluster contains the attributes shown in Table 39.
Table 39 applies to proxy devices.

Table 39 – Attributes of the GP client cluster

ID	Name	Type	Range	Access	Default	M/O	Description
0x0000-0x000f	Defined by the server side (A.3.3.2)						
0x0010	*gppMaxProxy-TableEntries*	unsigned 8-bit integer	Any valid	R	0x14	M	Maximum number of Proxy Table entries supported by this device
0x0011	*Proxy Table*	Long octet string	N/A	R	0x0000	M	Proxy Table, holding infor-mation about pairings between a particular GPD ID and the sinks in the network
0x0012	*gppNotificationRe-tryNumber*	unsigned 8-bit integer	0x00-0x05	R/W	0x02	X (M if *full unicast communi-cation* functional-ity supported)	Number of full unicast GP Notification retries on lack of GP Notification Response
0x0013	*gppNotificationRe-tryTimer*	unsigned 8-bit integer	0x00 – 0xff	R/W	0x64	X (M if *full unicast communi-cation* functional-ity supported)	Time in ms between full unicast GP Notification retries on lack of GP Notification Response
0x0014	*gppMaxSearch-Counter*	Unsigned 8-bit integer	Any valid	R/W	0x0a	X (O if *Proxy Table mainte-nance* function-ality supported)	The frequency of sink re-discovery for inactive Proxy Table entries
0x0015	*gppBlockedGPDID*	Long octet string	N/A	R	0x0000	X (O if *Proxy Table mainte-nance* function-ality supported)	A list holding information about blocked GPD IDs
0x0016	*gppFunctionality*	24-bit bitmap	N/A	R	Any valid	M	The optional GP functionality supported by this proxy
0x0017	*gppActiveFunctionali-ty*	24-bit bitmap	N/A	R	0xffffff	M	The optional GP functionality supported by this proxy that is active
0x0018 - 0x001f	Reserved for further Green Power cluster client side attributes						
0x0020 - 0x002f	Attributes shared by proxy and sink, as defined in Table 24						
0x0030 -0xffff	Reserved						

[134]With respect to ZCL Default Response handling for the ZCL foundation commands to manipulate the GP proxy attributes, the proxy SHALL follow section 2.5.12.2 of ZCL r06 or later (see [3]) and, in addition, for ZCL Write Attributes command, also section 2.5.3.3 of ZCL r06 or later (see [3]).

A.3.4.2.1 gppMaxProxyTableEntries attribute

Maximum number of Proxy Table entries this node can hold.

Any proxy type SHALL support at least five Proxy Table entries.

The recommended number of the Proxy Table entries for a Basic Proxy is twenty.

Note: in a system with sinks using broadcast GP Pairing commands, and all proxies storing information about all GPD, this limits the total number of the GPD to 5. If more GPDs need to be supported in a system, additional means can be used, e.g. bigger Proxy Tables can be implemented, some intelligence can be employed to limit the number of proxies forwarding on behalf of each GPD (e.g. by a sink or a Commissioning Tool) or Proxy Table maintenance functionality can allow for dynamic Proxy Table adaptation.

A.3.4.2.2 Proxy Table attribute

The Proxy Table attribute contains the information on GPDs active in the system and the corresponding sinks.

Proxy Table is a read-only attribute. Generic ZCL commands cannot be used to create/modify or remove *Proxy Table* entries. If required, e.g. for CT-based commissioning, the GP Pairing command of the Green Power cluster can be used for that purpose.

[135]The Proxy Table SHALL be persistently stored across restarts, OTA upgrades and power cycles.

Specifically, a Green Power Proxy Basic SHALL persistently store all mandatory parameters of a Proxy Table entry and all configured optional parameters of a Proxy Table entry, with the following exceptions:

- The Green Power Proxy Basic MAY, but is not required to, persistently store the *GPD security frame counter* parameter of the Proxy Table entry. Upon restart, the *GPD security frame counter* parameter SHALL have a value lower than or equal to the last value observed before restart.
- The Green Power Proxy Basic MAY, but is not required to, persistently store the following subfields of the *Options* parameter of the Proxy Table entry: *FirstToForward*, *InRange*, *HasAllUnicastRoutes*, since they are not used in any way by the Green Power Proxy Basic.

A.3.4.2.2.1 Over the air transmission of Proxy Table

When sent over the air in a ZCL command carrying the Proxy Table attribute, it is represented as a long octet string, which internally has the format of a sequence of structures. Then, it contains the 2B length field of the Long octet string data format – defining the total length of the attribute, and then the Proxy Table entries itself, each of which is a structure, formatted as shown in Table 40. For each of the entries, the presence of the optional parameters is indicated by the corresponding flag in the *Options* or *Security Options* parameter:

- The *GPD ID* and *Endpoint* parameter:
  - *ApplicationID* = 0b000 indicates the *GPD ID* parameter has the length of 4B and contains the GPD SrcID; the *Endpoint* field is absent.
  - *ApplicationID* = 0b010 indicates the *GPD ID* parameter has the length of 8B and contains the

[134] CCB #2336; Resolution added in 15-02014-009
[135] CCB #2470, #2471; resolution added in 15-02014-014

GPD IEEE address; the *Endpoint* field is present.
    - All values of *ApplicationID* other than 0b000 and 0b010 are reserved in the current version of the Green Power cluster specification.
- *GPD Assigned Alias* parameter SHALL be included if *AssignedAlias* = 0b1, it SHALL be omitted otherwise;
- The parameters *Security Options* and *GPD key* SHALL always all be included if the *SecurityUse* sub-field is set to 0b1 (irrespective of the key type in use); *SecurityUse* sub-field is set to 0b0, the parameters *Security Options*, and *GPD key* SHALL be omitted.
- *GPD security frame counter* parameter SHALL:
    - be present and carry the value of the *Security frame counter*, if:
        - *SecurityUse* = 0b1,
        - *SecurityUse* = 0b0 and *MAC sequence number capabil*ities = 0b1;
    - be omitted if *SecurityUse* = 0b0 and *Sequence number capabilities* = 0b0.
- *Lightweight sink address list* parameter
    - SHALL only be included if *Lightweight unicast GPS* sub-field of the *Options* parameter is set to 0b1;
      whereby the first octet indicates the number of entries in the list, and the entries of the list follow directly as defined in Table 41; no additional length/element number indication is included per entry;
    - SHALL be omitted completely otherwise (i.e. even the length octet SHALL be omitted);
- *Sink group list* parameter
    - SHALL only be included if *Commissioned Group GPS* sub-field of the *Options* parameter is set to 0b1;
      whereby the first octet indicates the number of entries in the list, and the entries of the list follow directly, formatted as defined in Table 26;
    - SHALL be completely omitted otherwise (i.e. even the length octet SHALL be omitted);
- [136]*Search Counter* SHALL be included if *EntryActive* or *EntryValid* sub-field of the *Options* parameter is set to 0b0, it SHALL be omitted otherwise;
- [137]*Extended Options* and *Full unicast sink address list* SHALL be omitted by all devices implemented according to the current specification; the *Options Extension* sub-field of the *Options* field SHALL be set to 0b0.

The proxy SHALL only respond with ZCL Read Attributes Response with Status = SUCCESS, if all configured Proxy Table entries fit completely into a single response frame (without fragmentation or partitioning cluster usage). Otherwise, the proxy SHALL respond with ZCL Read Attributes Response with Status = INSUFFICIENT_SPACE and no entries included. For the values of the Status codes see [3].

A.3.4.2.2.2 Proxy Table entry format

Implementers of this specification are free to implement the Proxy Table in any manner that is convenient and efficient, as long as it represents the data shown in Table 40.

[136] CCB #2275; Resolution added in 15-02014-006
[137] CCB #2275; Resolution added in 15-02014-006

Table 40 – Format of entries in the Proxy Table

Parameter name	Type	Range	Default	M / O	Description
Options	16-bit bitmap	Any valid	N/A	M	This parameter specifies the tunneling options
GPD ID	Unsigned 32-bit integer/IEEE address	Any valid	N/A	M	ID of the GPD
Endpoint	Unsigned 8-bit integer	0x01-9xf0, 0xff	N/A	O (M if *ApplicationID* = 0b010)	GPD endpoint
GPD Assigned Alias	Unsigned 16-bit integer	0x0001-0xfff7	N/A	O	The commissioned 16-bit ID to be used as alias for this GPD
Security Options	8-bit bitmap	Any valid	N/A	O (M if *Security use* = 0b1)	The security options
GPD security frame counter	Unsigned 32-bit Integer	Any valid	0xffffffff	O	The incoming security frame counter for the GPD
GPD key	Security key	Any valid	N/A	O	The security key for the GPD. It MAY be skipped, if common/derivable key is used (as indicated in the *Options* parameter)
Lightweight sink address list	sequence of octets	Any valid	0x00	O (M if *Lightweight unicast GPS* =0b1)	IEEE and short address of the sink(s) that requires tunneling in lightweight unicast communication mode
Sink group list	sequence of octets	Any valid	0x00	O (M if *Commissioned Group GPS*=0b1)	GroupIDs and Aliases for the sinks that require the tunneling in groupcast communication mode
Groupcast radius	Unsigned 8-bit integer	0x00 – 0xff	0xff	M	To limit the range of the groupcast
Search Counter	Unsigned 8-bit integer	0x00 - *gpp-MaxSearch-Counter*	0x00	O (M if *EntryActive*=0b0 or *EntryValid=0b0)*	For inactive/invalid entries, allows for Sink re-discovery when Search Counter equals *0*
Extended Options	16-bit bitmap	Any valid	N/A	O (M if *Options Extension* = 0b1)	This parameter specifies extensions to the tunneling options
Full unicast sink address list	sequence of octets	Any valid	0x00	O (M if *Full Unicast GPS* =0b1)	IEEE and short address of the sink(s) that requires tunneling in full unicast communication mode

Each proxy SHALL be able to support per Proxy Table entry, i.e. per GPD any of the following minimum configurations: (i) at least 2 entries in the *Lightweight sink address list* and/or *Full unicast sink address list*, (ii) at least 2 entries in the *Sink group list* and (iii) at least 1 entry in the *Lightweight sink address list* or *Full unicast sink address list* and at least 1 entry in the *Sink group list*.

A.3.4.2.2.2.1 Options parameter

The *Options* parameter SHALL be formatted as shown in Figure 66 and Figure 67.

Bits: 0..2	3	4	5	6	7	8	9
ApplicationID	EntryActive	EntryValid	Sequence number capabilities	Lightweight Unicast GPS	Derived Group GPS	Commissioned Group GPS	FirstToForward

Figure 66 – Format of the Options parameter of the Proxy Table entry (part 1)

Bits: 10	11	12	13	14	15
InRange	GPD Fixed	HasAllUni-castRoutes	AssignedAlias	SecurityUse	Options Exten-sion

Figure 67 – Format of the Options parameter of the Proxy Table entry (part 2)

The *ApplicationID* sub-field contains the information about the application used by the GPD. *ApplicationID* = 0b000 indicates the *GPD ID* parameter has the length of 4B and contains the GPD SrcID; the *Endpoint* field is absent. *ApplicationID* = 0b010 indicates the *GPD ID* parameter has the length of 8B and contains the GPD IEEE address; the *Endpoint* field is present. All values of *ApplicationID* other than 0b000 and 0b010 are reserved in the current version of the Green Power cluster specification.

The *EntryActive* sub-field, if set to 0b1, indicates, that the current Proxy Table entry is active. A Proxy Table entry with the *EntryActive* flag equal to 0b0 can contain the *SearchCounter* parameter.

The *EntryValid* sub-field, if set to 0b1, indicates, that the current Proxy Table entry contains complete sink information.

The *Sequence number capabilities* sub-field can have the values as defined in A.4.2.1.1.2.

The *Lightweight Unicast GPS* sub-field, if set to 0b1, indicates that there is at least one sink paired to this GPD, that requires lightweight unicast communication mode. Then, *Lightweight sink address list* parameter is present.

The *Derived Group GPS* sub-field, if set to 0b1, indicates that there is at least one sink paired to this GPD, that requires groupcast communication mode with automatically-derived DGroupID (see A.3.6.1.4).

The *Commissioned Group GPS* sub-field, if set to 0b1, indicates that there is at least one sink paired to this GPD, that require groupcast communication mode with the pre-commissioned GroupID.

The *FirstToForward* sub-field is a Boolean flag used for *gppTunnelingDelay* calculation.

The *InRange* sub-field, if set to 0b1, indicates that this GPD is in range of this proxy. The default value is FALSE.

The *GPDfixed* sub-field, if set to 0b1, indicates portability capabilities of this GPD. The default value is FALSE.

The *HasAllUnicastRoutes* sub-field, if set to 0b1, indicates that the proxy has active routes to all full unicast sinks for this GPD; if set to 0b0, it indicates that at least one full unicast route is missing.

The *AssignedAlias* sub-field, if set to 0b1, indicates that the assigned alias as stored in the *GPD Assigned Alias* parameter SHALL be used instead of the alias derived from the GPD ID (sec. A.3.6.3.3) in case of [138]full unicast and derived groupcast communication modes. If set to 0b0, the derived alias is used (sec. A.3.6.3.3) for those communication modes.

The *Security use* sub-field, if set to 0b1, indicates that security-related parameters of the Sink Table entry are present.

The *Options Extension* sub-field, if set to 0b1, indicates that the *Extended Options* field is present.

A.3.4.2.2.2.2 Endpoint field

The *Endpoint* field SHALL be present if *ApplicationID* = 0b010. It then carries the identifier of the GPD endpoint, which jointly with the GPD IEEE address identifies a unique logical GPD device. If *ApplicationID* = 0b000 the *Endpoint* field SHALL be absent.

[138] CCB #2397; Resolution added in 15-02014-010

The values 0xf1 - 0xfe are reserved for future use. The value 0x00 indicates application endpoint-independent communication and SHOULD be used e.g. for channel and key updates. The value 0xff indicates 'all endpoints'.

A.3.4.2.2.2.3 GPD Assigned Alias parameter

The *GPD Assigned Alias* parameter, if present – as indicated by the *AssignedAlias* sub-field of the *Options* field - , stores the assigned alias NWK source address to be used for this GPD in case of full unicast communication GPS or derived groupcast communication GPS, instead of the default alias derived from the GPD ID (sec. A.3.6.3.3).

Note: In case of lightweight unicast communication GPS, aliasing is not used. In case of commissioned groupcast communication GPS, the alias is stored in the Sink group list parameter, together with the corresponding pre-commissioned GroupID.

A.3.4.2.2.2.4 Security-related parameters

[139]The security-related parameters are formatted and SHALL be used as described inA.3.3.2.2.2.6.

A.3.4.2.2.2.5 Lightweight sink address list parameter

The entries in the *Lightweight sink address list* parameter SHALL have the format as specified in Table 41. It contains the list of paired lightweight unicast sinks for this GPD.

Table 41 – Format of entries in the *Lightweight sink address list* parameter of the Proxy Table

Parameter name	Type	Description
Sink IEEE address	IEEE address	IEEE address of the GP sinks which require the tunneling in unicast communication mode
Sink NWK address	Unsigned 16-bit integer	NWK short address matching the sink's IEEE address

A.3.4.2.2.2.6 Sink group list parameter

The *Sink group list* contains the list of sink GroupIDs for this GPD, with the corresponding aliases.

The entries in the *Sink group list* parameter SHALL be formatted as specified in Table 26.

If the *Pre-Commissioned Group GPS* sub-field of the *Options* parameter is set, the *Sink group list* SHOULD be present.

A.3.4.2.2.2.7 Groupcast radius parameter

The *Groupcast radius* contains the intended radius for the groupcast communication, in number of hops. The default value of 0x00 indicates unspecified, i.e. twice the value of the *nwkMaxDepth* attribute of the NIB, as specified by [1].

If *Groupcast radius* parameter is set to a value 0x00 and another value is received, the new value SHALL be kept. If *Groupcast radius* parameter is set to a value other than 0x00 and a new value is received, the higher value SHALL be kept.

A.3.4.2.2.2.8 Extended Options parameter

The *Extended Options* parameter SHALL be formatted as shown in Figure 68.

[139] CCB #2292; Resolution added in 15-02014-006

Bits: 0	1..15
Full unicast GPS	Reserved

Figure 68 – Format of the Extended Options parameter of the Proxy Table entry (part 1)

The *Full Unicast GPS* sub-field, if set to 0b1, indicates that there is at least one sink paired to this GPD, that requires full unicast communication mode. Then, *Full unicast sink address list* parameter is present.

A.3.4.2.2.2.9 Full unicast sink address list

The entries in the *Full unicast sink address list* parameter SHALL have the format as specified in Table 41. It contains the list of paired full unicast sinks for this GPD.

A.3.4.2.3 gppNotificationRetryNumber attribute

This attribute defines the maximum number of retransmissions in case a GP Notification Response command is not received from a particular sink for full unicast GP Notification command.

A.3.4.2.4 gppNotificationRetryTimer attribute

This attribute defines the time to wait for GP Notification Response command after sending full unicast GP Notification command.

A.3.4.2.5 gppMaxSearchCounter attribute

This attribute defines the maximum value the Search Counter can take, before it rolls over.

A.3.4.2.6 gppBlockedGPDID attribute

The *gppBlockedGPDID* attribute contains the information on GPDs active in the vicinity of the network node, but not belonging to the system.

It is a long octet string, which internally has the format of an array of structures. Thus, the ZCL command carrying the *gppBlockedGPDID* attribute contains the 2B length field of the Long octet string data format – defining the total length of the attribute; and then the entries of the *gppBlockedGPDID* itself; each of which is a structure, formatted as shown in Table 42.

Implementers of this specification are free to implement the *gppBlockedGPDID* in any manner that is convenient and efficient, as long as it represents the data shown in Table 42.

Table 42 – Format of entries in the gppBlockedGPDID attribute

Parameter name	Type	Range	Default	M / O	Description
Options	Unsigned 8-bit integer	Any valid	N/A	M	Options related to this list entry
GPD ID	Unsigned 32-bit integer/IEEE address	Any valid	N/A	M	ID of the GPD
Endpoint	Unsigned 8-bit integer	Any valid	N/A	O (M if *ApplicationID* = 0b010)	GPD Endpoint
Sequence number	Unsigned 8-bit integer	0x00-0xff	0x00	M	The last sequence number observed from this GPD.

Parameter name	Type	Range	Default	M / O	Description
Search Counter	Unsigned 8-bit integer	0x00 - *gpp-MaxSearch-Counter*	0x00	M	Allows for Sink re-discovery when Search Counter equals *0*

The *Options* parameter SHALL be formatted as shown in Figure 69.

Bits: 0..2	3..7
ApplicationID	Reserved

Figure 69 – Format of the Options parameter of the gppBlockedGPDID attribute entry

The *ApplicationID* sub-field contains the information about the application used by the GPD. *ApplicationID* = 0b000 indicates the GPD ID parameter has the length of 4B and contains the GPD SrcID; the *Endpoint* field is absent. *ApplicationID* = 0b010 indicates the GPD ID parameter has the length of 8B and contains the GPD IEEE address; the *Endpoint* field is present. All values of *ApplicationID* other than 0b000 and 0b010 are reserved in the current version of the Green Power cluster specification.

This parameter is an optimization, allowing for storing only limited information for the purpose of GPDF filtering. Equivalent information can be stored in the Proxy Table.

If supported, the *gppBlockedGPDID* attribute SHALL contain at least 10 entries.

A.3.4.2.7 gppFunctionality attribute

The *gppFunctionality* attribute indicates support of the GP functionality by this device. Any 1-bit sub-field set to 0b1 indicates that this functionality is supported; set to 0b0 indicates that this functionality is not implemented. The reserved sub-fields and sub-fields for any non-applicable functionality SHALL also be set to 0b0.

The *gppFunctionality* attribute is formatted as shown in Table 43.

The rightmost column shows the values used by the Basic Proxy, standalone or as part of Green Power Basic Combo.

Table 43 – Format of the gppFunctionality attribute

Indication	Functionality	Basic Proxy
b0	GP feature	0b1
b1	Direct communication (reception of GPDF via GP stub)	0b1
b2	Derived groupcast communication	0b1
b3	Pre-commissioned groupcast communication	0b1
b4	Full unicast communication	0b0
b5	Lightweight unicast communication	0b1
b6	Reserved	0b0
b7	Bidirectional operation	0b0
b8	Proxy Table maintenance (active and passive, for GPD mobility and GPP robustness)	0b0
b9	Reserved	0b0
b10	GP commissioning	0b1
b11	CT-based commissioning	0b1
b12	Maintenance of GPD (deliver channel/key during operation)	0b0

b13	gpdSecurityLevel = 0b00	0b1
b14	Deprecated: gpdSecurityLevel = 0b01	0b0
b15	gpdSecurityLevel = 0b10	0b1
b16	gpdSecurityLevel = 0b11	0b1
b17	Reserved	0b0
b18	Reserved	0b0
b19	GPD IEEE address	0b1
[140]b20	Reserved	0b0
[141]b21 – b23	Reserved	0b0

For all Green Power Proxy, Green Power Basic Proxy and proxy functionality of Green Power combo or Green Power Basic Combo, the following sub-fields SHALL always be set as follows:

- b0 = 0b1 (M functionality);
- b1 = 0b1 (M functionality);
- b6 = 0b0 (N/A functionality);
- b9 = 0b0 (N/A functionality);
- b17 = 0b0 (N/A functionality);
- b18 = 0b0 (N/A functionality);
- b20 = 0b0 (N/A functionality)[142].

A.3.4.2.8 gppActiveFunctionality attribute

The *gppActiveFunctionality* attribute indicates which GP functionality supported by this device is currently enabled. Any 1-bit sub-field set to 0b1 indicates that this functionality is supported and enabled; set to 0b0 indicates that this functionality is disabled or not implemented.

The *gppActiveFunctionality* attribute is formatted as shown in Table 29.

The *GP feature* sub-field of the *gppActiveFunctionality* attribute is a master flag. By writing 0b1/0b0 to the *GP feature* sub-field, the complete GP operation can be enabled/disabled, respectively. Even when the *GP feature* sub-field is set to 0b0, the GP attributes SHALL be accessible and the Simple Descriptor for the Green Power EndPoint SHALL be readable.

In the current version of the GP specification, the *gpsActiveFunctionality* attribute is read only, and the *GP feature* sub-field SHALL be set to 0b1.

In the current version of the GP specification, the remaining sub-fields of the *gpsActiveFunctionality* attribute are reserved and SHALL be set to 0b1. If future version of the GP specification would define further *gpsActiveFunctionality* flags, they SHOULD be aligned with *gpsFunctionality* attribute.

[140] CCB #2418, resolved in 15-02014r010
[141] CCB #2418, resolved in 15-02014r010
[142] CCB #2418, resolved in 15-02014r010

A.3.4.3 Commands received

Whether the support of particular command is mandatory or optional is dependent on the GP infrastructure device type and the functionality it supports, and specified in Table 23.

Table 44 – Green Power cluster: client side: commands received

Command ID	Command Name	Command Description	Link
0x00	GP Notification Response	From sink to a proxy to acknowledge GP Notification received in full unicast mode.	A.3.3.5.1
0x01	GP Pairing	From sink to proxies to (de)register for tunneling service or to remove GPD from the network.	A.3.3.5.2
0x02	GP Proxy Commissioning Mode	From sink to proxies in the whole network to indicate commissioning mode.	A.3.3.5.3
0x03-0x05	Reserved		
0x06	GP Response	From sink to selected proxies, to provide data to be transmitted to Rx-capable GPD.	[143]A.3.3.5.4
0x07	Reserved		
0x08	Reserved		
0x09	Reserved		
0x0a	GP Sink Table Response	To receive information on requested selected Sink Table entries, by index or by GPD ID	A.3.3.5.6
0x0b	GP Proxy Table Request	To request selected Proxy Table entries, by index or by GPD ID	A.3.4.3.1
0x0c – 0xff	Reserved		

A.3.4.3.1 GP Proxy Table Request command

The payload of the GP Proxy Table Request command SHALL be formatted as illustrated in Figure 70.

Octets	1	0/4/8	0/1	0/1
Data Type	8-bit bitmap	unsigned 32-bit integer/IEEE address	unsigned 8-bit integer	unsigned 8-bit integer
Field Name	Options	GPD ID	Endpoint	Index

Figure 70 – Format of the GP Proxy Table Request command

The *Options* field of the GP Proxy Table Request command is formatted as shown in Figure 71.

Bits: 0..2	3..4	5..7
ApplicationID	Request type	Reserved

Figure 71 – Format of the Options field of the GP Proxy Table Request command

[143] CCB #2416, resolved in 15-02014r010

The *ApplicationID* sub-field contains the information about the application used by the GPD. *ApplicationID* = 0b000 indicates the *GPD ID* field, if present as indicated by the *Request type* sub-field of the *Options* field, has the length of 4B and contains the GPD SrcID; the *Endpoint* field is absent. *ApplicationID* = 0b010 indicates the *GPD ID* field, if present as indicated by the *Request type* sub-field of the *Options* field, has the length of 8B and contains the GPD IEEE address; the *Endpoint* field is present. All values of *ApplicationID* other than 0b000 and 0b010 are reserved in the current version of the Green Power cluster specification.

The *Request type* sub-field specifies how table entries are requested. It SHALL take one of the non-reserved the values defined in Table 36.

If set to 0b00, it indicates that the *GPD ID* (and *Endpoint* field, is *ApplicationID* = 0b010) field is present and carries the GPD ID (and *Endpoint* field, is *ApplicationID* = 0b010) for which the Proxy Table entry is requested; the *Index* field is absent.

If set to 0b01, indicates that the *Index* field is present and carries the starting index for the Proxy Table entry request; the *GPD ID* and *Endpoint* fields are absent.

The *GPD ID* field carries the value of the *GPD ID*, either GPD SrcID or GPD IEEE address, depending on the value of the *ApplicationID*, for which the Proxy Table entry is requested.

The *Endpoint* field carries the value of the GPD endpoint for which the Proxy Table entry is requested.

The *Index* field carries the index value of the Proxy Table entry being requested. The index enumeration includes only non-empty Proxy Table entries. It starts with 0x00; 0xff indicates unspecified.

A.3.4.3.1.1 When generated

The GP Proxy Table Request command is generated to read out selected Proxy Table entry(s), by index or by GPD ID.

If the sender of the command wishes to avoid receiving many responses, esp. from the nodes not supporting this functionality, it SHALL set the *Disable default response* sub-field of the *Frame Control* field of the ZCL header of the GP Proxy Table Request command, as specified in sec. 2.3.1.1.4 of [3].

A.3.4.3.1.2 Effect on receipt

On receipt of this command, the device is informed about a request for selected Proxy Table entries.

A.3.4.4 Commands generated

Whether the support of particular command is mandatory or optional is dependent on the GP infrastructure device type and the functionality it supports, and specified in Table 23.

4085 Table 45 – Green Power cluster: client side: commands generated

Command ID	Command Name	Command Description	Link
0x00	GP Notification	From proxy to sink(s) to tunnel GP frame.	A.3.3.4.1
0x01	GP Pairing Search	From proxy to the sinks in entire network to get pairing indication related to GPD for Proxy Table update.	A.3.3.4.2
0x02	Reserved		
0x03	GP Tunneling Stop	From proxy to neighbor proxies to indicate GP Notification sent in full unicast mode.	A.3.4.4.1
0x04	GP Commissioning Notification	From proxy to sink(s) to tunnel GPD commissioning data.	A.3.3.4.3
0x05	Reserved		
0x06 – 0x09	Reserved		
0x0a	GP Sink Table Request	To request selected Sink Table entries	A.3.3.4.7
0x0b	GP Proxy Table Re-sponse	To send selected Proxy Table entries	A.3.4.4.2
0x0c-0xff	Reserved		

4086

A.3.4.4.1 GP Tunneling Stop command

The payload of the GP Tunneling Stop command SHALL be formatted as illustrated in Figure 72.

Octets	1	4/8	0/1	4	2	1
Data Type	8-bit bitmap	unsigned 32-bit integer/IEEE address	unsigned 8-bit integer	unsigned 32-bit integer	unsigned 16-bit integer	8-bit bitmap
Field Name	Options	GPD ID	Endpoint	GPD security frame counter	GPP short address	GPP-GPD link

Figure 72 – Format of the GP Tunneling Stop command

The *Options* field of the GP Tunneling Stop command SHALL be formatted as illustrated in Figure 73.

Bits: 0..2	3	4		5..7
ApplicationID	Also Derived Group	Also Commissioned Group		Reserved

Figure 73 – Format of the Options field of the GP Tunneling Stop command

The *ApplicationID* sub-field contains the information about the application used by the GPD. *ApplicationID* = 0b000 indicates the GPD ID field has the length of 4B and contains the GPD SrcID; the *Endpoint* field is absent. *ApplicationID* = 0b010 indicates the GPD ID field has the length of 8B and contains the GPD IEEE address; the *Endpoint* field is present. All values of *ApplicationID* other than 0b000 and 0b010 are reserved in the current version of the Green Power cluster specification.

The flags *Also Derived Group* and *Also Commissioned Group*, if set to 0b1, indicate presence of sinks paired to the same GPD with a different communication mode.

The *GPD ID* field has the value copied from the GPDF *SrcID* field/GPDF MAC header *Source address* field, depending on the value of the *ApplicationID* in the GPDF.

The *Endpoint* field has the value copied from the GPDF *Endpoint* field.

The *GPD security frame counter* field is always present. If the *SecurityLevel* sub-field of the *Extended NWK Frame Control* field of the received GPDF was 0b00, it carries the value copied from the GPDF MAC header *Sequence number* field, pre-padded with 0x000000. Otherwise, if the *SecurityLevel* sub-field of the *Extended NWK Frame Control* field of the received GPDF was 0b10- 0b11, it carries the value copied from the *Security frame counter* field of the received GPDF that was successfully used for the security processing of the received GPDF.

The fields *GPP address* and *GPP-GPD link* are always present and carry the short address of the originating proxy, and the quality of the received GPDF, as reported by the dGP-DATA.indication primitive, respectively. The *GPP-GPD link* field of the GP Tunneling Stop command is formatted as shown in Figure 27 and calculated as defined in sec. A.3.3.4.1.

The *Disable default response* sub-field of the *Frame Control Field* of the ZCL header SHALL be set to 0b1.

A.3.4.4.1.1 When generated

This command is sent to prevent other proxies from also forwarding GP Notifications to the sinks requiring full unicast communication mode.

A.3.4.4.1.2 Effect on Receipt

On receipt of this command, a device is informed about another proxy forwarding a GPDF.

A.3.4.4.2 GP Proxy Table Response command

The GP Proxy Table Response command SHALL be formatted as illustrated in Figure 74.

Octets	1	1	1	1	0/Variable	…	0/Variable
Data Type	8-bit enumeration	Unsigned 8-bit integer	unsigned 8-bit integer	unsigned 8-bit integer	Octet string	…	Octet string
Field Name	Status	Total number of non-empty Proxy Table entries	Start index	Entries count	Proxy Table entry	,,,	Proxy Table entry

Figure 74 – Format of the GP Proxy Table Response command

The *Status* field can take the values of SUCCESS or NOT_FOUND (for the values of the Status codes see [3]).

The *Total number of non-empty Proxy Table entries* field specifies the total number of non-empty Proxy Table entries currently available on the responding device. Value of 0x00 indicates the Proxy Table is empty. Value of 0xff indicates Proxy Table is not implemented.

The *Start index* field specified the table position of the first of the Proxy Table entry included. The first non-empty entry in the Proxy Table has *Index* value 0.

The *Entries count* field specifies the number of *Proxy Table entry* fields included in the current message.

Each *Proxy Table entry* field contains a complete Proxy Table entry, formatted as specified in sec. A.3.4.2.2.1. The entries are ordered by *Index* field value, with the lowest entry being sent first.

A.3.4.4.2.1 When generated

Upon reception of the GP Proxy Table Request command, the device SHALL check if it implements a Proxy Table.

If not, it SHALL generate a ZCL Default Response command, with the *Status code* field carrying UNSUP_CLUSTER_COMMAND, subject to the rules as specified in sec. 2.4.12 of [3].

If the device implements the Proxy Table, it SHALL prepare a GP Proxy Table Response.

If its Proxy Table is empty, and the triggering GP Proxy Table Request was received in unicast, then the GP Proxy Table Response SHALL be sent with *Status* [144]NOT_FOUND, *Total number of non-empty Proxy Table entries* carrying 0x00, *Start index* carrying 0xFF (in case of request by GPD ID) or the *Index* value from the triggering GP Sink Table Request (in case of request by index), *Entries count* field set to 0x00, and any *Proxy Table entry* fields absent.

If the triggering GP Proxy Table Request command contained an *Index* field, the device SHALL check if it has at least Index+1 non-empty Proxy Table entries. If not, the device SHALL create a GP Proxy Table Response with *Status* NOT_FOUND, *Total number of non-empty Proxy Table entries* carrying the total number of non-empty Proxy Table entries on this device, *Start index* carrying the *Index* value from the triggering GP Proxy Table Request, *Entries count* field set to 0x00 and any *Proxy Table entry* fields absent. If yes, the device SHALL create a GP Proxy Table Response with *Status* SUCCESS, *Total number of non-empty Proxy Table entries* carrying the total number of non-empty Proxy Table entries on this device, *Start index* carrying the *Index* value from the triggering GP Proxy Table Request, *Entries count* field set to the number of complete Proxy Table entries, which are included, followed by those *Proxy Table entry* fields themselves, formatted as specified in sec. A.3.4.2.2.1.

[144] CCB #2171; Resolution added in 15-02014-005

Note: the device SHALL only include complete Proxy Table entries; if an entry does not fit completely into the frame, it SHALL NOT be included in this response.
Note 2: If there are empty Proxy Table entries between non-empty Proxy Table entries, they SHALL NOT be included in the response.

If the triggering GP Proxy Table Request command contained a *GPD ID* field, the device SHALL check if it has a Proxy Table entry for this GPD ID (and *Endpoint*, if *ApplicationID* = 0b010). If yes, the device SHALL create a GP Proxy Table Response with *Status* SUCCESS, *Total number of non-empty Proxy Table entries* carrying the total number of non-empty Proxy Table entries on this device, *Start index* set to 0xff, *Entries count* field set to 0x01, and one *Proxy Table entry* field for the requested GPD ID (and *Endpoint*, if *ApplicationID* = 0b010), formatted as specified in sec. A.3.4.2.2.1, present.

If the entry requested by GPD ID (and *Endpoint*, if *ApplicationID* = 0b010) cannot be found, and the triggering GP Proxy Table Request was received in unicast, then the GP Proxy Table Response SHALL be sent with *Status* NOT_FOUND, *Total number of non-empty Proxy Table entries* carrying the total number of non-empty Proxy Table entries on this device, *Start index* carrying 0xFF, *Entries count* field set to 0x00, and any *Proxy Table entry* fields absent. If the triggering GP Proxy Table Request was received in groupcast or broadcast, then the GP Proxy Table Response SHOULD be skipped.

A.3.4.4.2.2 Effect on receipt

On receipt of this command, the remote device is informed about selected Proxy Table entries on the sending device.

A.3.5 Green Power operation

A.3.5.1 Overview

The proxies forward the Data GPDFs from the GPDs to paired sinks as regular Zigbee messages using the ZCL Green Power cluster commands.

Each sink has as part of the Green Power cluster a Sink Table to store pairing information between GP devices and its bound local application endpoints.

As a result of the commissioning actions, the sink manages the entries in its Sink Table. Sink Table entry changes for a particular GPD are announced to the proxies by sending a GP Pairing command. The sink responds to the proxies' GP Pairing Search commands requesting missing information on paired GPDs by sending GP Pairing commands.

Each sink is responsible for mapping and translating the received GP application commands of the paired GPDs into proper ZCL commands, and executing them properly. If the received GP application command requires bidirectional communication, and the requesting GPD is RxAfterTx-capable, the sink forms the response and sends it to the device it has selected for sending the response to the GPD.

Each proxy has as part of the Green Power cluster a Proxy Table to store pairing information on the GPDs and the paired sinks, including the security requirements and communication mode.

The proxy participates in management of pairings at the sinks, by switching between commissioning and operational mode upon reception of GP Proxy Commissioning Mode command and, when in commissioning mode by tunneling the received GPD commissioning data even for unknown GPDs as regular Zigbee messages using the ZCL Green Power cluster GP Commissioning Notification command. On receipt of GP Pairing command frames, the proxy manages the entries in its Proxy Table. The proxy can ask for updates on missing or outdated pairing information by sending GP Pairing Search command.

The proxy is responsible for tunneling the received Data GPDFs of the GPDs for which it has valid pairing information to the paired sink, as the regular Zigbee messages using the ZCL Green Power cluster GP Notification command.

The proxy forwards Data GPDF to an RxAfterTx-capable GPD, if requested by the sink as indicated by GP Response command.

A.3.5.2 Description

A.3.5.2.1 Green Power Proxy (GPP) operation

On receipt of GP-SEC.request, the proxy acts as described in sec. A.3.7.3.1.1.

On receipt of Zigbee Update Device and Device_annce commands with IEEE address other than 0xffffffffffffffff, the proxy SHALL check if it has the announced device listed in the *SinkAddressList* of its Proxy Table. If yes, the mapping of the Sink IEEE address to the Sink NWK address SHALL be updated. Further, the proxy SHALL check if the NWKAddr field matches any of the aliases used by this proxy. If that's the case, an address conflict is with a regular Zigbee device is discovered and the proxy SHALL act according to Zigbee [1] address conflict announcement procedure, i.e. the proxy SHALL send after randomly chosen delay from between Dmin and Dmax (see A.3.6.3.1) the Zigbee Device_annce command (unless identical frame was received within this time), formatted as described in sec. A.3.6.3.4.2, to force the regular Zigbee device to change its short address. The alias SHALL NOT be changed.

On receipt of GP Proxy Commissioning Mode command, the proxy enters or exits the commissioning mode, according to the value of the *Action* sub-field of the *Options* field. It also adapts other parameters, e.g. *Channel*, *ExitMode* and *CommissioningWindow* duration, according to the values received in the GP Proxy Commissioning Mode command. It further exits the commissioning mode, when the exit conditions specified in the *ExitMode* sub-field of the previously received GP Proxy Commissioning Mode command are fulfilled (see Figure 22) or when *CommissioningWindow* times out. If the *ExitMode* had the *On first Pairing success* sub-field set to 0b1, the proxy SHALL exit commissioning mode upon reception of any GP Pairing command, including GP Pairing command with *RemoveGPD* sub-field set to 0b1 or *AddSink* sub-field set to 0b0.

On receipt of GP Pairing command in commissioning mode, the proxy updates its Proxy Table, if the entry is active.

Note: if *ApplicationID* = 0b010, the *Endpoint* field of a Proxy Table entry for a GPD IEEE address has either the exact value as the *GPD Endpoint* field in the incoming message, or 0xff.

If the *RemoveGPD* sub-field of the *Options* field was set to 0b0 and the *SecurityLevel* field of the *Options* field is set to 0b01, the proxy SHALL NOT update (if existent) nor create a Proxy Table entry.

If the *RemoveGPD* sub-field was set to 0b1, the proxy, if it does not support the *Proxy Table maintenance* functionality, SHALL remove the Proxy Table entry for that GPD; if the *ApplicationID* = 0b010 and the value of the *Endpoint* field of the GP Pairing command is other than 0xff, the proxy SHALL remove that entry, if existing; if the *ApplicationID* = 0b010 and the value of the *Endpoint* field of the GP Pairing command is 0xff, the proxy SHALL remove all entries for this GPD IEEE address. If the proxy does support the *Proxy Table maintenance* functionality, it SHALL either set this entry to inactive valid instead, if supported, or shift it to *gppBlockedGPDID* list, if implemented.

If the *RemoveGPD* sub-field was set to 0b0; and the *AddSink* sub-field was set to 0b0, the proxy removes the sink's address or Sink group address from the *SinkList*, depending on the setting of the *CommunicationMode* sub-field. If the removed unicast/group sink address is the last in the *Lightweight* or *Full unicast sink address list*/*Sink group list*, respectively, and no other sink communication mode is used for this entry, then the proxy proceeds as follows. If the proxy supports the *Proxy Table maintenance* functionality, the proxy SHALL set the entry status to inactive valid or shift it to *gppBlockedGPDID* list, if implemented; the SearchCounter SHALL be set to 0x00. If the proxy does not support the *Proxy Table maintenance* functionality, the proxy SHALL remove this Proxy Table entry.

If the *RemoveGPD* sub-field was set to 0b0 and the *AddSink* sub-field was set to 0b1, the proxy adds the communication mode, if new, and the sink (group) address, if not already included in the *SinkList* to this entry, and sets this entry to active and valid. If a groupcast sink is being added to a Proxy Table entry, the proxy also adds its Green Power EndPoint as a member of the specified group. The proxy updates the Proxy Table fields *SecurityLevel*, *KeyType*, *GPDkey* and *GPDsecurityFrameCounter*, if they were included in the GP Pairing command; if *ApplicationID* = 0b010, the proxy SHALL check if it has another entry for the same GPD IEEE address and update the security fields. If the *Assigned Alias* field is present, the proxy stores it in the relevant Proxy Table entry, and sets the corresponding *Options* sub-field.

[145]Furthermore, on receipt of GP Pairing command with *RemoveGPD* flag was set to 0b0 and the *AddSink* flag was set to 0b1, the proxy [146]MAY check if the supplied alias, derived or assigned, is identical with the proxy's own short address. If it is, address conflict is discovered and the proxy SHALL act according to Zigbee [1] address conflict resolution procedure, i.e. the proxy SHALL randomly choose a new short address and subsequently announce it using the Zigbee Device_annce command short address. The alias SHALL NOT be changed.

On receipt of GP Pairing command in operational mode, the proxy checks if it has an active valid Proxy Table entry for this GPD. If yes, the proxy performs the changes to this entry, as requested by the GP Pairing command. The proxy SHALL NOT send Device_annce for the alias. It is assumed, that the Device_annce is sent by the sink or CT sending the GP Pairing command. If the *RemoveGPD* sub-field of the *Options* field was set to 0b0 and the *SecurityLevel* field of the *Options* field is set to 0b01, the proxy SHALL NOT update (if existent) nor create a Proxy Table entry.

[147]On receipt of a GP Response frame from the sink, both in operational and commissioning mode, the proxy checks if either (i) the GP Response was sent to the proxy in groupcast and its short address matches the value in the *TempMaster short address* field or (ii) the GP Response command was sent to this proxy in unicast. If yes, the proxy adds the GPDF frame derived from the GP Response frame to its *gpTxQueue* for sending to the indicated GPD ID (and *Endpoint*, if *ApplicationID* = 0b010) by calling GP-DATA.request with *Action* parameter set to TRUE with bit5 of the *TxOptions* set to the value of the *Tx on matching endpoint* sub-field of the *Options* field of the GP Response command, and sets its *FirstToForward* flag for this GPD to 0b1.
If the *TempMaster* short address field of the GP Response command carries an address different than the short address of the receiving proxy, the proxy drops the current command, sets the *FirstToForward* flag for the relevant Proxy Table entry to 0b0, and proceeds as follows. If *ApplicationID* sub-field of the GP Response command is set to 0b000, the proxy removes any previous pending GPDF for this GPD from its *gpTxQueue* by calling GP-DATA.request with the *Action* parameter set to FALSE, and sets the *FirstToForward* flag for this SrcID in its Proxy Table to 0b0. If *ApplicationID* sub-field of the GP Response command is set to 0b010, the proxy instructs the dGP stub to remove pending relevant GPDF for this GPD IEEE address (see sec. A.1.3.2.3) from its *gpTxQueue* by calling GP-DATA.request with the *Action* parameter set to FALSE, bit5 of the *TxOptions* set to the value of the *Tx on matching endpoint* sub-field of the *Options* field of the GP Response command, and the GPD IEEE address and GPD Endpoint copied from the GP Response; and sets the *FirstToForward* flag for this GPD in its Proxy Table to 0b0.

[145] CCB #2408; resolution added in 15-02014-010
[146] CCB #2408; resolution modified in 15-02014-013 as a result of Kavi comment #1378 from letter ballot for GP Bsic errata set: https://workspace.zigbee.org/higherlogic/ws/groups/PRO_GP/comments/view_comment?comment_id=1378
[147] CCB #2379; resolution added in 15-02014-011

On receipt of GP-DATA.indication, the proxy checks the GPDF type and the mode the proxy is in.

If the proxy is in operational mode, and the GPDF carries a correctly protected GPD Commissioning or GPD Decommissioning command from a GPD the proxy has a Proxy Table entry for, the proxy SHALL forward the GPD command to the paired sinks using GP Notification command in the appropriate communication mode(s).

[148]If the proxy is in operational mode, and the GPDF carries a correctly protected GPD Success command [149]or any other GPD commissioning command from the range 0xE4 – 0xEF, from a GPD the proxy has a Proxy Table entry for, the proxy SHOULD NOT forward the GPD command using the GP Notification; however, if generated, GP Notification command SHALL be sent to the paired sinks using command in the appropriate communication mode(s).

If the proxy is in operational mode, and the GPDF carries a GPD Commissioning command, GPD Success command, GPD Channel Request,a GPD Decommissioning command [150]or any other GPD commissioning command from the range 0xE4 – 0xEF, from a GPD the proxy has no Proxy Table entry for, or incorrectly protected GPDF from a GPD the proxy has a Proxy Table entry for, the frame [151]SHALL be silently dropped.

If the GPDF carries a Decommissioning GPDF, and the proxy is in commissioning mode, and the GP-DATA.indication had the Status of SECURITY_SUCCESS or NO_SECURITY, the proxy updates the *GPD security frame counter* parameter of the relevant Proxy Table entry for this GPD and schedules sending of GP Commissioning Notification. If GP-DATA.indication had the Status of AUTH_FAILURE, the proxy MAY schedule transmission of GP Commissioning Notification, with the *Security processing* flag set to 0b1.

If the GPDF is a Commissioning GPDF or a Data GPDF with *Auto-Commissioning* flag set to 0b1 and the proxy is in commissioning mode, the proxy acts as described in sec. A.3.9.1.

If the GP-DATA.indication Status is SECURITY_SUCCESS/NO_SECURITY and the GPDF is a Data GPDF, independent of whether the *Auto-Commissioning* flag is set to 0b0 or 0b1, and the proxy is in operational mode, the proxy searches its Proxy Table for a matching entry related to the received GPD ID (and any *Endpoint*, if *ApplicationID* = 0b010). If there is any active Proxy Table entry for this GPD ID with the *InRange* flag set to 0b0 (even if the *GPDfixed* flag is also set to 0b1 or if the *Endpoint* field has value other than in the received GPDF), the Proxy sets the *InRange* flag to 0b1. Then, the proxy continues as follows.

If *ApplicationID* = 0b010, the proxy checks if it has a Proxy Table entry with *GPD IEEE address* and the *Endpoint* parameter set either to the exact value from the GPDF or to 0xff. If not, the GPDF is silently dropped.

If an entry exists and the entry is active and valid then the proxy checks the security level of the received GPDF as follows. The proxy compares the value of the sub-fields *SecurityLevel* and *SecurityKey* from for the received GPDF command with the corresponding *SecurityLevel* and *SecurityKey* parameters from the Proxy Table. If the *SecurityLevel* and the *SecurityKey* do match, the proxy performs freshness check (see sec. A.3.6.1.2.1). If any of those checks fails and on reception of GP-DATA.indication with the Status AUTH_FAILURE or UNPROCESSED, the proxy stops processing the frame. The proxy SHALL NOT send GP Tunneling Stop/GP Notification; it MAY send GP Pairing Search.

[148] CCB #2517; resolution added in 15-02014-014
[149] CCB #2517; resolution added in 15-02014-015
[150] CCB #2517; resolution added in 15-02014-015
[151] CCB #2517; resolution added in 15-02014-014

If all the checks succeed, the proxy stores the *Sequence Number / Frame Counter* in the *GPD security frame counter parameter* of this Proxy Table entry, and constructs from the received GPDF a GP Notification command(s) for each communication mode stored in the Proxy Table for this GPD; if *ApplicationID* = 0b010, the *Endpoint* field of the GP Notification command SHALL be set to the value of the *Endpoint* field from the triggering GPDF. If the *RxAfterTx* sub-field of the received GPDF was set to 0b1, the *RxAfterTx* sub-field of the *Options* field SHALL be set to 0b1, the *BidirectionalCommunicationCapability* sub-field SHALL be set according to device capabilities, and the *gpTxQueueFull* sub-field of the *Options* field SHALL be set according to the status of this proxy's *gpTxQueue* (i.e., if there is no entry in the *gpTxQueue* for this GPD and the queue is full, it sets the *gpTxQueueFull* sub-field to 0b1, otherwise if it has an entry for this GPD or at least one empty entry, it sets it to 0b0); if the proxy does not support bidirectional communication, it SHALL set the *gpTxQueueFull* sub-field of the *Options* field to 0b1. The GPD *CommandID* and *GPD Command payload* are included in the clear in the GP Notification command, even if they were encrypted in the GPDF (*SecurityLevel* = 0b11); the *MIC* field from the GPDF SHALL NOT be included. The lower layers of the proxy stack (APS and NWK layer of Zigbee) will take care of appropriate protection of the command during tunneling through the Zigbee network. The *Ack. request* sub-field of the APS *Frame Control* field is set to 0b0.

If the proxy is not capable of bidirectional communication or if the *RxAfterTx* sub-field of the *Extended NWK Control Field* of the triggering GPDF was set to 0b0, for groupcast GP Notification, the proxy SHALL further use the following values: NWK Src address = alias source address (see A.3.6.3.3); NWK Sequence Number = alias sequence number (see A.3.6.3.3); NWK Dest address: 0xFFFD (broadcast to RxOnWhenIdle=TRUE); APS group address: as stored in the Proxy Table, APS source endpoint: Green Power EndPoint, APS counter: alias sequence number (see A.3.6.3.3).

If the proxy is capable of bidirectional communication and the *RxAfterTx* sub-field of the *Extended NWK Control Field* of the triggering GPDF was set to 0b1, for groupcast GP Notification, the proxy SHALL further use the following values: NWK Src address, NWK sequence number and APS counter: proxy's own values (no aliasing), NWK Dest address: 0xFFFD (broadcast to RxOnWhenIdle=TRUE); APS group address: as stored in the Proxy Table, APS source endpoint: Green Power EndPoint.

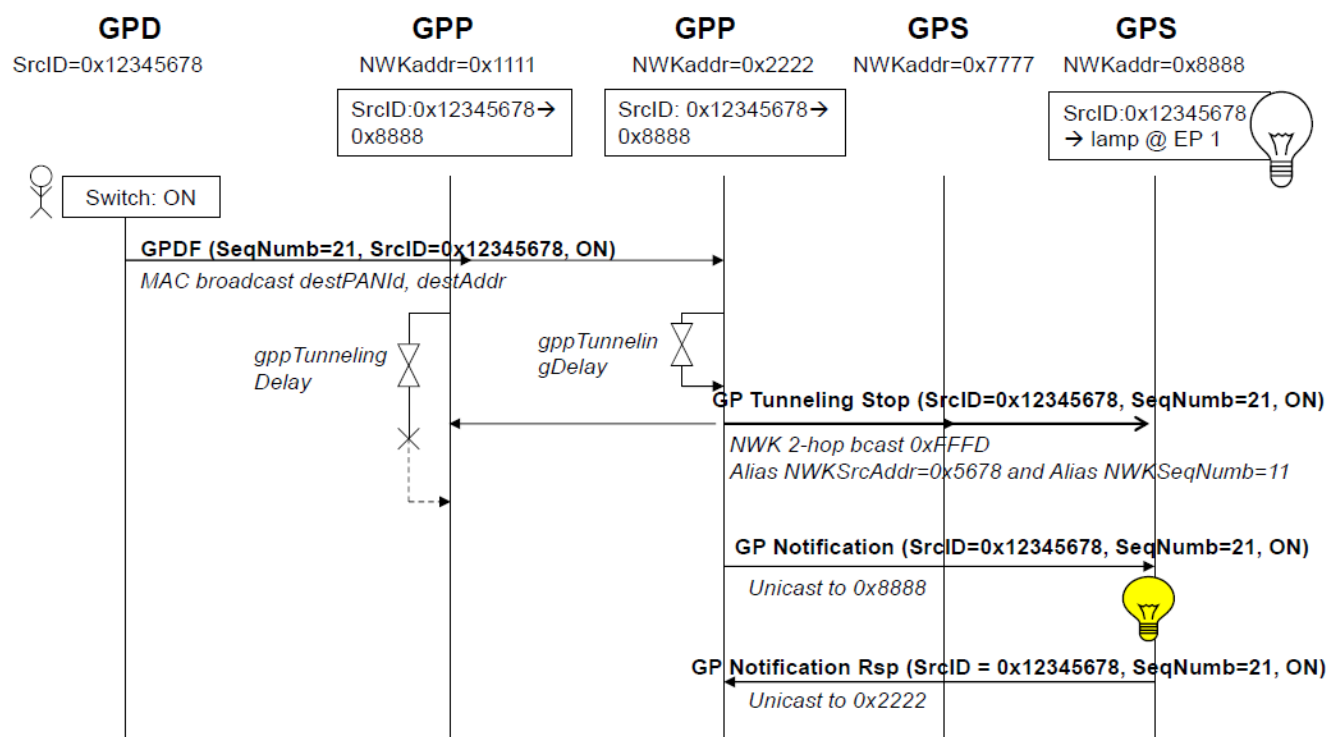
For the full and lightweight unicast GP Notification command, the proxy SHALL further use the following values: NWK Src address, NWK sequence number and APS counter: proxy's own values (no aliasing), NWK Dst address: sink's short address, APS source and destination end point: Green Power EndPoint. For the GP Tunneling Stop command the proxy SHALL use proxy aliasing (see sec. A.3.6.3.3) for NWK Src address, NWK Sequence Number, and APS Counter; local radius (2 hops), and 0xFFFD broadcast as NWK Dest address.

The proxy schedules sending of the GP Notification command. If (i) there are only lightweight unicast destinations and/or (ii) groupcast destinations, and the *RxAfterTx* flag was cleared, the sending SHALL be scheduled after *Dmin* (see section A.3.6.3.1). Otherwise, if (i) there are any full unicast destinations, also in addition to groupcast destinations, or (ii) the *RxAfterTx* flag was set, the sending SHALL be scheduled after *gppTunnelingDelay* (see section A.3.6.3.1); if there are full unicast destinations, the *gppTunnelingDelay* is calculated as for the full unicast. If the proxy is capable of bidirectional communication or there are any full unicast destinations, and during *gppTunnelingDelay* the proxy receives a GP Tunneling Stop, or a GP (Commissioning) Notification related to the GPDF scheduled for tunneling, it SHALL drop all the scheduled transmissions resulting from the same GPDF, if the *RxAfterTx* flag was set to 0b0. Otherwise, if the *RxAfterTx* flag was set to 0b1, the proxy SHALL only drop the scheduled transmissions, if the *BidirectionalCommunicationCapability* sub-field of the *Options* field was set to 0b1 and either *GPP-GPD link* field from the received command has a better value than measured by the receiving proxy on receipt of this GPDF (whereby better *GPP-GPD link* is defined as one having higher value of the *Link quality* sub-field, and if *Link quality* is equal, as one having higher value of the *RSSI* sub-field), or if the *GPP-GPD link* value is equal and the value in the *GPP address* field of the received GP Tunneling Stop/GP (Commissioning) Notification is lower than this proxy's NWK address.

On *gppTunnelingDelay* / *Dmin* timeout, respectively, the GP Tunneling Stop command (if any) SHALL be sent first, the remaining commands SHOULD be sent in the following order: the lightweight or full unicast GP Notification(s) (if any), groupcast GP Notification(s) (if any). Upon transmission of full unicast GP Notification, the proxy SHALL wait for *gppNotificationRetryTimer* ms for a GP Notification Response, and re-transmits upon its lack, up to *gppNotificationRetryNumber* times. If GP Notification Response command is received, the scheduled (re-)transmissions of the GP Notification command to this sink are dropped, and the *FirstToForward* bit in the proxies' Proxy Table entry for this GPD (and the indicated/0xff *Endpoint*, if *ApplicationID* = 0b010) is updated, taking the value in GP Notification Response as input. If the *NoPairing* flag of the GP Notification Response command is set to 0b1, the proxy SHALL remove this sink from its *SinkAddressList* in the Proxy Table entry for this GPD (and the indicated/0xff *Endpoint*, if *ApplicationID* = 0b010). If no GP Notification Response command is received after last retry of the full unicast GP Notification, the proxy MAY request the Zigbee stack to re-discover the route to this full unicast sink. It MAY pro-actively clear the *HasAllUnicastRoutes* sub-field of the *Options* parameter of the Proxy Table entry for this GPD (and the indicated/0xff *Endpoint*, if *ApplicationID* = 0b010).

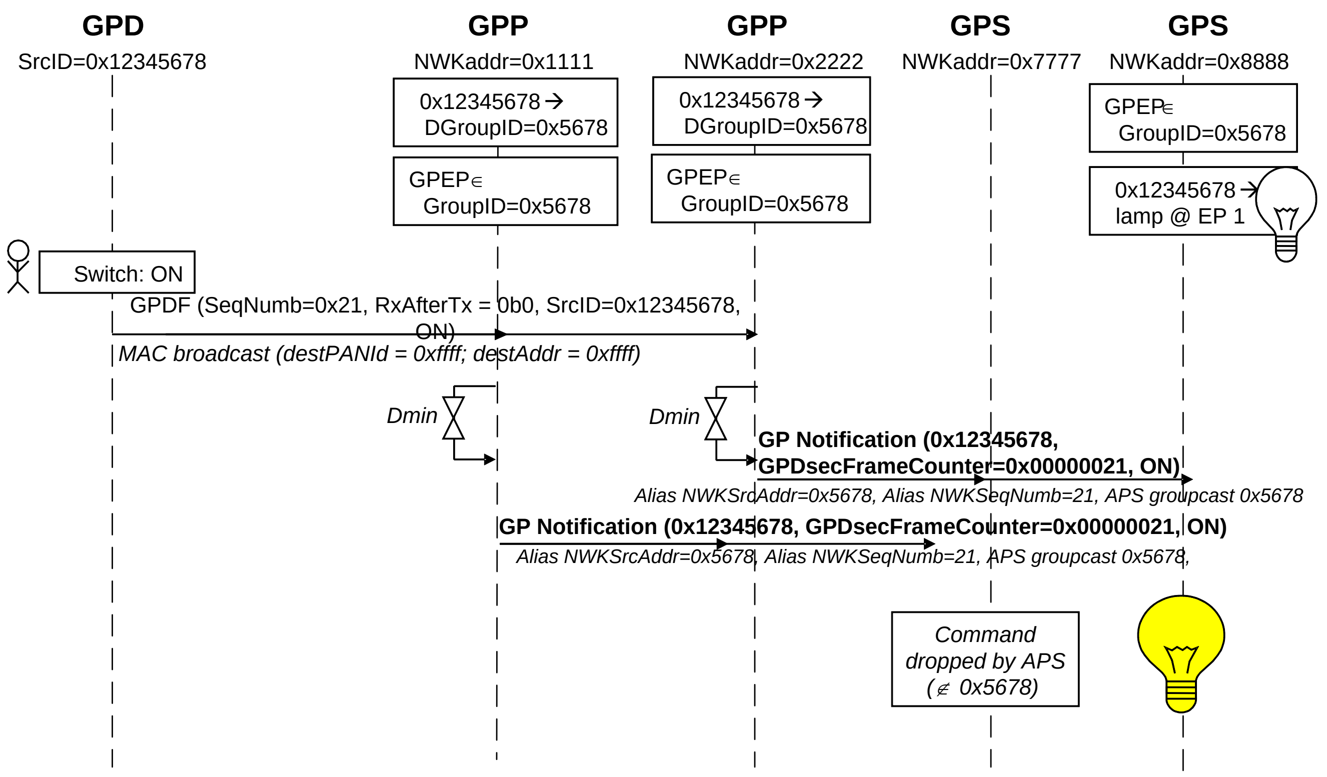
For groupcast communication, the proxy sets the *FirstToForward* sub-field of the Proxy Table entry itself to 0b1, if it managed to forward the GP Notification frame, and to 0b0 otherwise. When there are many paired sinks for the same GPD ID (and matching *Endpoint*, if *ApplicationID* = 0b010), the proxy uses the OR function for setting the *FirstToForward* flag in its Proxy Table entry, i.e. if the *FirstToForward* is set in at least one GP Notification Response, and/or the proxy manages to send at least one groupcast GP Notification, it sets the *FirstToForward* flag in its Proxy Table.

Exemplary message sequence charts are depicted in Figure 75 and Figure 76.


GPD
SrcID=0x12345678
GPP
NWKaddr=0x1111
GPP
NWKaddr=0x2222
GPS
NWKaddr=0x7777
GPS
NWKaddr=0x8888
SrcID:0x12345678→ 0x8888
SrcID: 0x12345678→ 0x8888
SrcID:0x12345678 → lamp @ EP 1
Switch: ON
GPDF (SeqNumb=21, SrcID=0x12345678, ON)
MAC broadcast destPANId, destAddr
gppTunneling Delay
gppTunnelin gDelay
GP Tunneling Stop (SrcID=0x12345678, SeqNumb=21, ON)
NWK 2-hop bcast 0xFFFD
Alias NWKSrcAddr=0x5678 and Alias NWKSeqNumb=11
GP Notification (SrcID=0x12345678, SeqNumb=21, ON)
Unicast to 0x8888
GP Notification Rsp (SrcID = 0x12345678, SeqNumb=21, ON)
Unicast to 0x2222

4412 **Figure 75 – Exemplary message sequence chart for GPD with SrcID for Green Power full unicast communication**

4413


GPD
SrcID=0x12345678
GPP
NWKaddr=0x1111
GPP
NWKaddr=0x2222
GPS
NWKaddr=0x7777
GPS
NWKaddr=0x8888
0x12345678→ DGroupID=0x5678
GPEP∈ GroupID=0x5678
0x12345678→ DGroupID=0x5678
GPEP∈ GroupID=0x5678
GPEP∈ GroupID=0x5678
0x12345678→ lamp @ EP 1
Switch: ON
GPDF (SeqNumb=0x21, RxAfterTx = 0b0, SrcID=0x12345678, ON)
MAC broadcast (destPANId = 0xffff; destAddr = 0xffff)
Dmin
Dmin
GP Notification (0x12345678, GPDsecFrameCounter=0x00000021, ON)
Alias NWKSrcAddr=0x5678, Alias NWKSeqNumb=21, APS groupcast 0x5678
GP Notification (0x12345678, GPDsecFrameCounter=0x00000021, ON)
Alias NWKSrcAddr=0x5678, Alias NWKSeqNumb=21, APS groupcast 0x5678,
Command dropped by APS (∉ 0x5678)

4415 **Figure 76 – Exemplary message sequence chart for GPD with SrcID for GP groupcast communication when**
4416 **RxAfterTx = 0b0**

The proxy behavior in the following situations will be defined by the application profile: (i) on receipt of unsolicited GP Pairing command in operational mode when there is no Proxy Table entry (ii) on receipt of GP Pairing command in commissioning mode when there is no Proxy Table entry, (iii) GP Notification forwarding on receipt of Data GPDF in commissioning mode.

In sec. A.3.7.3.2, SDL diagrams for the above described operation are provided.

A.3.5.2.2 Proxy Table maintenance

If the *Proxy Table maintenance* functionality is supported, it SHALL be implemented in the following way.

The proxy can passively discover the information by storing pairing information from GP Notification and GP Tunneling Stop commands sent by other proxies, both in operational and commissioning mode. Active discovery is performed by sending GP Pairing Search or broadcast GP Notification command. Appropriate Proxy Table entry status allows avoiding too many discovery broadcasts. For example, keeping inactive entries for GPD nodes without a pairing in the network allows avoiding repetitive pairing re-discovery (with the resource-consuming network-wide broadcast of the GP Pairing Search command). It can be used e.g. for keeping information on GPDs in a neighbor network or on GPDs removed from the network.

A.3.5.2.2.1 Proxy Table entry status

The proxy can store entries with different status values in its Proxy Table. The entry status as a function of the *EntryActive* and *EntryValid* flags is explained in Table 46.

Table 46 – Proxy Table entry status

EntryActive	EntryValid	Meaning
1	1	(According to this proxy's knowledge) The GPD with this GPD ID (and *Endpoint*, if *ApplicationID* = 0b010) belongs to this Zigbee network, the sink information is current and valid.
1	0	(According to this proxy's knowledge) The GPD with this GPD ID (and *Endpoint*, if *ApplicationID* = 0b010) belongs to this Zigbee network, the sink information MAY be outdated/incomplete/not available (e.g. because it just restarted).
0	0	(According to this proxy's knowledge) The GPD with this GPD ID (and *Endpoint*, if *ApplicationID* = 0b010) does not belong to this Zigbee network, though this information MAY be outdated/wrong (e.g. because it just restarted).
0	1	(According to this proxy's knowledge) The GPD with this GPD ID (and *Endpoint*, if *ApplicationID* = 0b010) does not belong to this Zigbee network (anymore), and this information is valid (e.g. because GP Pairing with *RemoveGPD* was received).

Alternatively, the inactive valid or inactive invalid entries of the Proxy Table can be moved into *gppBlockedGPDID* attribute, with the relevant information preserved (GPDID, Endpoint, Sequence number, SearchCounter).

A.3.5.2.2.2 Maintenance

[152]The proxy stores the pairing information persistently. On restart, the proxy SHOULD set the *EntryValid* flag of its Proxy Table entries to 0b0 and clear the *FirstToForward* and *HasAllUnicastRoutes* flags; it SHALL keep the sink address information. Subsequently, the proxy SHOULD rediscover its inactive Proxy Table entries. The proxy MAY perform Proxy Table read-out (see A.3.5.2.2.6) or Active re-discovery (see A.3.5.2.2.5). If GP Pairing Search command is sent, it SHALL have the *Request GPD Security Frame Counter* flag set to 0b1.

[152] CCB #2470, #2471; resolution added in 15-02014-014

On receipt of GP Pairing command, the proxy SHALL always check its Proxy Table, both in commissioning and operational mode. The proxy SHALL NOT send Device_annce for the alias. It is assumed, that the Device_annce is sent by the sink or CT sending the GP Pairing command.
If the proxy has no Proxy Table entry for this GPD (and the indicated/0xff *Endpoint*, if *ApplicationID* = 0b010), it SHOULD create a new active valid entry, especially if the *FixedLocation* flag is set to 0b0 or if the *FixedLocation* flag is set to 0b1 and the proxy is in the radio range of this GPD; and store all GPD capability information available from GP Pairing.
On receipt of a GP Pairing with *RemoveGPD* flag set to 0b1, rather than removing the Proxy Table entry, the proxy SHALL set its Proxy Table entry for this GPD (and the indicated/0xff *Endpoint*, if *ApplicationID* = 0b010) to inactive and valid; all sink flags [153](i.e. sub-fields *Lightweight Unicast GPS, Derived Group GPS, Commissioning Group GPS* of the *Options* field and *Full Unicast GPS* of the *Extended Options* field of the Proxy Table entry) SHALL be cleared and all sinks removed.
If the Proxy Table entry becomes empty, i.e. if its *Lightweight or Full unicast sink address list* contains an address of a single sink, and the proxy receives a GP Pairing command from this sink with the *AddSink* bit in the *Options* field set to 0b0 or if its *Sink group list* contains a single GroupID and the proxy receives a GP Pairing command for this group, with the *AddSink* sub-field in the *Options* field set to 0b0, the proxy SHALL perform Active re-discovery (see sec.A.3.5.2.2.5).
If the proxy receives a GP Pairing command with *AddSink* set to 0b1 for an inactive and valid entry, it SHALL store the supplied pairing information and set the status to active valid.
If the proxy receives a GP Pairing command with *AddSink* set to 0b1 for an invalid entry, it SHALL store the supplied pairing information and set the status to active valid; it SHOULD also perform active re-discovery (see A.3.5.2.2.5).

On receipt of GP-DATA.indication for Data GPDF with Status AUTH_FAILURE or UNPROCESSED in operational mode, with a GPD ID (and *Endpoint*, if *ApplicationID* = 0b010) for which the proxy has active invalid Proxy Table, it SHALL drop the frame and SHALL NOT send GP Tunneling Stop/GP Notification.

On receipt in operational mode of a GP-SEC.request for Data GPDF, for an inactive and valid entry, the proxy returns GP-SEC.response with Status DROP_FRAME; the *SearchCounter* is incremented.
On receipt of a GP Tunneling Stop or a GP Notification for an inactive and valid entry, the command is silently dropped and no further action is taken.

On receipt of GP-DATA.indication for Data GPDF with Status SECURITY_SUCCESS in operational mode, with a GPD ID for which the proxy does not have Proxy Table entry, the proxy creates an active invalid entry for this GPD, sets the Search counter to 0, the *InRange* flag to 0b1, and performs Passive discovery (see A.3.5.2.2.3). The proxy MAY also derive the DGroupID and add its Green Power EndPoint as a member of this group in its *apsGroupTable*. If *ApplicationID* = 0b010, and the proxy already has an entry for the GPD IEEE address and one particular endpoint (not equal to 0xff), the proxy MAY create active invalid entry for the received *Endpoint*, as described above, and proceed with Passive discovery (see A.3.5.2.2.3).

[153] CCB #2280; Resolution added in 15-02014-006

On receipt in operational mode of GP-DATA.indication for Data GPDF with Status AUTH_FAILURE or UNPROCESSED or NO_SECURITY, with a GPD ID for which the proxy does not have Proxy Table entry, the proxy creates an inactive invalid entry for this GPD, sets the Search counter to 0, the *InRange* flag to 0b1, and performs Passive discovery (see A.3.5.2.2.3); if *ApplicationID* = 0b010, it MAY also be done in the case when the proxy already has an entry for the GPD IEEE address and one particular endpoint not equal to 0xff. The proxy MAY also derive the DGroupID and add its Green Power EndPoint as a member of this group in its *apsGroupTable*.

On receipt of GP-DATA.indication with Status SECURITY_SUCCESS in operational mode, with a GPD ID (and *Endpoint* matching or 0x00 or 0xff, if *ApplicationID* = 0b010) for which the proxy has active invalid Proxy Table, the proxy SHALL perform the checks as described in A.3.5.2.1. If any of the checks fail, the proxy SHOULD silently drop the frame. If the checks are successful, the proxy SHALL schedule transmission of broadcast GP Notification command after Dmin, the destination endpoint SHALL be set to 0xf2; the derived alias (see sec. A.3.6.3.3) SHALL be used if available in the Proxy Table entry; if the derived alias is not available, any of the assigned aliases can be used. If the entry for this GPD already contains sink information, the proxy SHALL NOT schedule transmission of GP Notification to the paired sinks in the requested communication mode. Then, the proxy proceeds as described in Active discovery (see sec. A.3.5.2.2.4).

If security processing of the Data GPDF in operational mode for an active valid Proxy Table entry fails, the proxy SHOULD send GP Pairing Search command with the *Request GPD Security Key* sub-field set to 0b1, if the *KeyType* is other than NWK key.

On receipt of a GP (Commissioning) Notification command or a GP Tunneling Stop command, for which the proxy has not seen the corresponding GPFS, the proxy SHALL check the content of its Proxy Table. If the entry for this GPD (and *Endpoint* matching or 0x00 or 0xff, if *ApplicationID* = 0b010) exists, the proxy clears the *FirstToForward* flag and the *InRange* flag in the *Options* field of the corresponding Proxy Table entry. Furthermore, if the Proxy Table entry is active and the proxy is in operational mode, it acts as follows. If the entry is active and valid, but the sink data in it is not consistent with the content of the received command, or if the entry is active and invalid, the proxy MAY perform Proxy Table read-out (see A.3.5.2.2.6) or Active re-discovery (see A.3.5.2.2.5). If at exiting the commissioning mode, a new Proxy Table entry does not include any sink address, group or individual, but does have at least one sink flag set to 0b1, the proxy marks the entry as inactive invalid, sets Search counter 0, and performs Active re-discovery.

Keeping *Sequence number* values in the *gppBlockedGPDID* entries MAY allow for entry status arbitration between the proxies.

A.3.5.2.2.3 Passive discovery

The proxy waits for *gppDiscoveryDelay*. If within this time the proxy receives:

- a GP Pairing Search or broadcast GP Notification for the same GPD ID (and matching *Endpoint*, if *ApplicationID* = 0b010) and communication modes, then it stops the gppDiscoveryDelay timer and performs Active discovery.
- a GP Tunneling Stop command for this GPD ID (and matching *Endpoint*, if *ApplicationID* = 0b010); if the *Also Derived Group* and/or the *Also Commissioned Group* flag of the GP Tunneling Stop command was set to 0b1, it sets the *DerivedGroupGPS* and/or the *CommissionedGroupGPS*, sub-field, respectively, of the *Options* parameter of the Proxy Table entry for GPD to 0b1, and then

performs Active re-discovery.

- a GP Pairing command for this GPD ID (and *Endpoint*, matching or 0x00 or 0xff, if *ApplicationID* = 0b010), then it sets the entry as active and invalid, stores the information received and performs Active re-discovery.
- a unicast/groupcast GP Notification command for this GPD ID (and *Endpoint*, if *ApplicationID* = 0b010), then it and adds the communication mode “groupcast with derived GroupID” to the corresponding Proxy Table entry. If at least one of the “also unicast/commissioned group” bits in the GP Notification command is set, the proxy SHALL perform Active re-discovery. If neither of these flags is set, the entry is set to active and valid; no further action is taken.
- neither a GP Pairing Search command, nor a GP Pairing command, nor a broadcast GP Notification command for this GPD ID (and *Endpoint*, if *ApplicationID* = 0b010), then the proxy acts as follows.

If on *gppDiscoveryDelay* expiration, the Proxy Table entry is:

- active, the proxy forwards the received frame using a GP Notification command in broadcast[154], and performs Active discovery.
- inactive and the SearchCounter equals 0, the proxy performs Active re-discovery.
- inactive and the SearchCounter differs from 0, the proxy increments the counter by 1 (and sets it to 0 if it had its maximum value), and no further action is taken.

A.3.5.2.2.4 Active discovery

The proxy initiates a timer with *gppDiscoveryDuration*. If at least one GP Pairing command for this GPD ID (and *Endpoint*, matching or 0x00 or 0xff, if *ApplicationID* = 0b010) with *AddSink* = 0b1 is received within *gppDiscoveryDuration*, this Proxy Table entry is marked as active and valid, and data from each such GP Pairing command is stored. Otherwise, if at *gppDiscoveryDuration* this Proxy Table entry does not include any sink address, group or individual, this Proxy Table entry is marked as inactive and invalid, and the Search counter is incremented by 1. If GP Pairing command with *AddSink* = 0b0 or *RemoveGPD* = 0b1 is received, the proxy acts as described in sec. A.3.5.2.1.

A.3.5.2.2.5 Active re-discovery

The proxy broadcasts a GP Pairing Search command. If the proxy entered this procedure because it had seen a GP Notification command, or if the *DerivedGroupGPS* sub-field of the *Options* parameter of the Proxy Table entry for GPD is set to, it SHALL clear the *Request Default Groupcast Sinks* sub-field in the GP Pairing Search command; the other two sink request sub-field are set, depending on the value of the corresponding flags in the triggering command. I.e., if the proxy entered this procedure because it had seen a GP Tunneling Stop command, it SHALL set the *Request unicast sinks* sub-field. The *Request Commissioned groupcast sinks* flag is set according to the value of the corresponding flag in the GP Tunneling Stop command or GP Notification command.

Then, the proxy starts a timer for gppDiscoveryDuration ms. If any GP Pairing command for this GPD ID (and *Endpoint*, matching or 0x00 or 0xff, if *ApplicationID* = 0b010) with *AddSink* = 0b1 is received within gppDiscoveryDuration, the Proxy Table entry for this GPD is marked as active and valid, and the data from each such GP Pairing command is stored. Otherwise, if no GP Pairing command with *AddSink* = 0b1 is received, at gppDiscoveryDuration expiration, the status of the Proxy Table entry remains unchanged, and - in case the Proxy Table entry is inactive– the Search counter is incremented by 1 (and set 0 if it had its maximum value). If GP Pairing command with *AddSink* = 0b0 or *RemoveGPD* = 0b1 is received, the proxy acts as described in sec. A.3.5.2.1.

[154] In this way, the command sent by the GPD is executed with the delay anticipated by the user. The GP Notification can in this case be seen as an implicit Pairing Search command: sink requiring other communication modes will send a GP Pairing command, cf. section A.2.4.3.1.2.

A.3.5.2.2.6 Proxy Table read-out

The proxy MAY read out interesting Proxy Table entries of other proxy, if any. A broadcast GP Notification SHALL NOT trigger the Proxy Table read-out.

The input SHALL only be used, if the read-out entry at the remote proxy is active and valid. Moreover, if the entry on the requesting proxy is also active and valid, it is recommended to only add sink information from the remote proxy.

A.3.5.2.2.7 gppDiscoveryDelay

The gppDiscoveryDelay is a constant, equal to the sum of Dmin, Dmax and 10 ms.

A.3.5.2.2.8 gppDiscoveryDuration

The gppDiscoveryDuration is a constant, equal to10s.

A.3.5.2.3 Operation of GP Proxy Basic and proxy side of GP Combo Basic

On receipt of GP-SEC.request, the Basic Proxy acts as described in sec. A.3.7.3.1.1.

On receipt of Zigbee Update Device and Device_annce commands with IEEE address other than 0xffffffffffffffff, the Basic Proxy SHALL check if it has the announced device listed in the *SinkAddressList* of any of its Proxy Table entries. If yes, the mapping of the Sink IEEE address to the Sink NWK address SHALL be updated. Further, the proxy SHALL check if the NWKAddr field of the received Device_annce matches any of the aliases used by this proxy. If that's the case, an address conflict with a regular Zigbee device is discovered and the proxy SHALL act according to Zigbee [1] address conflict announcement procedure, i.e. the proxy SHALL send, after randomly chosen delay from between Dmin and Dmax (see A.3.6.3.1), , the Zigbee Device_annce command (unless identical frame was received within this time), formatted as described inA.3.6.3.4.2, to force the regular Zigbee device to change its short address. The alias SHALL NOT be changed.

On receipt of GP Proxy Commissioning Mode command, the proxy enters or exits the commissioning mode, according to the value of the *Action* sub-field of the *Options* field. It also adapts other parameters, e.g. *Channel*, *ExitMode* and *CommissioningWindow* duration, according to the values received in the GP Proxy Commissioning Mode command. It further exits the commissioning mode, when the exit conditions specified in the *ExitMode* sub-field of the previously received GP Proxy Commissioning Mode command are fulfilled (see Figure 22) or when *CommissioningWindow* times out. If the *ExitMode* had the *On first Pairing success* sub-field set to 0b1, the proxy SHALL exit commissioning mode upon reception of any GP Pairing command, including GP Pairing command with *RemoveGPD* sub-field set to 0b1 or *AddSink* sub-field set to 0b0. While in commissioning mode, the Basic Proxy SHALL behave as described in sec. A.3.9.1, according to the supported commissioning functionality.

On receipt of GP Pairing command, the Basic Proxy updates its Proxy Table, as instructed by the GP Pairing command, both in commissioning and operational mode. The proxy SHALL NOT send Device_annce for the alias. It is assumed, that the Device_annce is sent by the sink or CT sending the GP Pairing command.

A received GP Pairing command with *GPD ID* field carrying SrcID = 0x00000000 (if *ApplicationID* = 0b000) or GPD IEEE address 0x0000000000000000 (if *ApplicationID* = 0b010) SHALL be silently dropped; Proxy Table entry SHALL NOT be created or updated. GP Pairing command with SrcID = 0xffffffff (if *ApplicationID* = 0b000) or GPD IEEE address 0xffffffffffffffff (if *ApplicationID* = 0b010) denotes a pairing for all GPD with a particular *ApplicationID* and SHALL be created if there is space in the Proxy Table.

[155]If the *GPD ID* field of a received GP Pairing command carries SrcID from the valid range 0x00000001 – 0xfffffff8 (if *ApplicationID* = 0b000) or GPD IEEE address from the valid range (if *ApplicationID* = 0b010), the proxy SHALL proceed as follows. If in the received GP Pairing command both *AddSink* sub-field of the *Options* field and *RemoveGPD* sub-field of the *Options* field are set to 0b1, the command SHALL be silently dropped, Proxy Table entries SHALL NOT be modified. [156]If *AddSink* sub-field of the *Options* field is set to 0b1 and the proxy has no Proxy Table entry for this GPD (and the indicated/0x00/0xff *Endpoint*, if *ApplicationID* = 0b010), the proxy SHALL check the *CommunicationMode* sub-field of the *Options* field. If the proxy does not support this *CommunicationMode* and the GP Pairing command was received in unicast, the proxy SHALL respond with ZCL Default response command with *Status* INVALID_FIELD; if the GP Pairing command was received in broadcast, the proxy SHALL silently drop it. If the proxy does support this *CommunicationMode*, it SHOULD create a new active valid entry, especially if the *FixedLocation* flag is set to 0b0 or if the *FixedLocation* flag is set to 0b1 and the proxy is in the radio range of this GPD; and store all GPD capability information available from GP Pairing. If the entry could not be created due to a lack of capacity in the Proxy Table, and the GP Pairing command was received in unicast, the proxy SHALL respond with ZCL Default response command with *Status* INSUFFICIENT_SPACE; if the GP Pairing command was received in broadcast, the proxy SHALL silently drop it.

If *AddSink* sub-field of the *Options* field is set to 0b1 and the proxy already has the Proxy Table entry for this GPD (and the indicated/0x00/0xff *Endpoint*, if *ApplicationID* = 0b010), it SHALL store the additional unicast or groupcast sink information, if any, in this Proxy Table entry, if there is still space. On receipt of a GP Pairing with *RemoveGPD* sub-field set to 0b1, the Basic Proxy SHALL remove this Proxy Table entry entirely. On receipt of a GP Pairing with *AddSink* sub-field of the *Options* field is set to 0b0 and *RemoveGPD* sub-field set to 0b0, if the proxy already has the Proxy Table entry for this GPD (and the indicated/0x00/0xff *Endpoint*, if *ApplicationID* = 0b010), it SHALL remove the indicated unicast or groupcast sink information, if stored, from this Proxy Table entry. If the Proxy Table entry becomes empty, i.e. if its *Lightweight or Full unicast sink address list* contains an address of a single sink, and the proxy receives a GP Pairing command from this sink with the *AddSink* bit in the *Options* field set to 0b0 or if its *Sink group list* contains a single GroupID and the proxy receives a GP Pairing command for this group, with the *AddSink* sub-field in the *Options* field set to 0b0, the proxy SHALL remove the entry entirely. If the proxy receives a GP Pairing command with *AddSink* set to 0b1 for an inactive and valid entry, it SHALL store the supplied pairing information and set the status to active valid. If the proxy receives a GP Pairing command with *AddSink* set to 0b1 for an invalid entry, it SHALL store the supplied pairing information and set the status to active valid; it SHOULD also perform active re-discovery (see A.3.5.2.2.5).

Note: if *ApplicationID* = 0b010, the *Endpoint* field of a Proxy Table entry for a GPD IEEE address has either the exact value as the *GPD Endpoint* field in the incoming message, or 0x00, or 0xff.

If the *RemoveGPD* sub-field of the *Options* field was set to 0b0 and the *SecurityLevel* field of the *Options* field is set to 0b01, the proxy SHALL NOT update (if existent) nor create a Proxy Table entry. If the *RemoveGPD* sub-field was set to 0b1, the Basic Proxy removes this Proxy Table entry; if the *ApplicationID* = 0b010 and the value of the *Endpoint* field of the GP Pairing command is other than 0xff, the proxy SHALL remove that entry, if existing; if the *ApplicationID* = 0b010 and the value of the *Endpoint* field of the GP Pairing command is 0xff, the proxy SHALL remove all entries for this GPD IEEE address. If the *RemoveGPD* sub-field was set to 0b0; and the *AddSink* flag was set to 0b0, the Basic Proxy removes the Sink group address from the *SinkGroupList*.

[155] CCB #2360; Resolution added in 15-02014-011
[156] CCB #2279 and CCB #2278; Resolution added in 15-02014-006

If the *RemoveGPD* sub-field was set to 0b0 and the *AddSink* flag was set to 0b1, the Basic Proxy adds the communication mode, if new, and the sink (group) address, if not already included in the corresponding *SinkList*, and sets the entry to active and valid. The Basic Proxy updates the Proxy Table fields *SecurityLevel*, *KeyType*, *GPDkey* and *GPDsecurityFrameCounter*, if they were included in the GP Pairing command; if *ApplicationID* = 0b010, the proxy SHALL check if it has another entry for the same GPD IEEE address and update the security fields. If the *Assigned Alias* field is present, the Basic Proxy stores it in its Proxy Table entry, and sets the corresponding *Options* sub-field.

[157]Furthermore, on receipt of GP Pairing command with *RemoveGPD* flag was set to 0b0 and the *AddSink* flag was set to 0b1, the proxy [158]MAY check if the supplied alias, derived or assigned, is identical with the proxy's own short address. If it is, address conflict is discovered and the proxy SHALL act according to Zigbee [1] address conflict resolution procedure, i.e. the proxy SHALL randomly choose a new short address and subsequently announce it using the Zigbee Device_annce command short address. The alias SHALL NOT be changed.

[159]

On receipt of GP-DATA.indication, the proxy checks the type of GPD command and the mode the proxy is in.

If the proxy is in operational mode, and SrcID = 0x00000000 (if *ApplicationID* = 0b000) or GPD IEEE address 0x0000000000000000 (if *ApplicationID* = 0b010), the frame SHALL be silently dropped. If the proxy is in operational mode, and the GPDF carries a correctly protected GPD Commissioning or GPD Decommissioning command from a GPD the proxy has a Proxy Table entry for, the proxy SHALL forward the GPD command to the paired sinks using GP Notification command in the appropriate communication mode(s); the *RxAfterTx* sub-field of the *Extended NWK Control Field* of the triggering GPDF SHALL be ignored, and the *RxAfterTx* sub-field of the *Options* field of the resulting GP Notification command SHALL always be set to 0b0; the GP Notification, if in groupcast, SHALL be sent with alias at *Dmin_u*. [160]If the proxy is in operational mode, and the GPDF carries a correctly protected GPD Success command [161]or any other GPD commissioning command from the range 0xE4 – 0xEF, from a GPD the proxy has a Proxy Table entry for, the proxy SHOULD NOT forward the GPD command using the GP Notification; however, if generated, GP Notification command SHALL be sent to the paired sinks using command in the appropriate communication mode(s); the *RxAfterTx* sub-field of the *Extended NWK Control Field* of the triggering GPDF SHALL be ignored, and the *RxAfterTx* sub-field of the *Options* field of the resulting GP Notification command SHALL always be set to 0b0; the GP Notification, if in groupcast, SHALL be sent with alias at *Dmin_u*. If the proxy is in operational mode, and the GPDF carries a GPD Commissioning command, GPD Success command, GPD Channel Request, a GPD Decommissioning command, [162]or any other GPD commissioning command from the range 0xE4 – 0xEF, from a GPD the proxy has no Proxy Table entry for, or incorrectly protected GPDF from a GPD the proxy has a Proxy Table entry for, the packet [163]SHALL be silently dropped. Otherwise, if the Basic Proxy is in commissioning mode, the Basic Proxy SHALL process the packet as described in sec. A.3.9.1.

[157] CCB #2408; resolution added in 15-02014-010
[158] CCB #2408; resolution modified in 15-02014-013 as a result of Kavi comment #1378 from letter ballot for GP Bsic errata set: https://workspace.zigbee.org/higherlogic/ws/groups/PRO_GP/comments/view_comment?comment_id=1378
[159] CCB #2279 and CCB #2278; Resolution added in 15-02014-006
[160] CCB #2517; resolution added in 15-02014-014
[161] CCB #2517; resolution added in 15-02014-015
[162] CCB #2517; resolution added in 15-02014-015
[163] CCB #2517; resolution added in 15-02014-014

If the GP-DATA.indication Status is SECURITY_SUCCESS/NO_SECURITY and the GPDF is a Data GPDF, independent of whether the *Auto-Commissioning* flag is set to 0b0 or 0b1, the Basic Proxy searches its Proxy Table for a matching entry related to the received GPD ID (and any *Endpoint*, if *ApplicationID* = 0b010). If there is any Proxy Table entry for this GPD with the *InRange* flag set to 0b0 (even if the *GPDfixed* flag is also set to 0b1 or if the *Endpoint* field has value other than in the received GPDF), the Basic Proxy sets the *InRange* flag to 0b1.

Then, the Basic Proxy continues as follows.

If *ApplicationID* = 0b010, the proxy checks if it has an entry with the exact *GPD ID* and *Endpoint* as in the GPDF, or otherwise if it has an entry with the exact *GPD ID* as in the GPDF and *Endpoint* = 0xff, or – if the Endpoint in the GP-DATA.indication is 0xff or 0x00 - if it has an entry with the exact *GPD ID*. If not, the GPDF is silently dropped.

If there is a matching entry, the Basic Proxy checks the security level of the received GPDF as follows. The Basic Proxy compares the value of the sub-fields *SecurityLevel* and *SecurityKey* from for the received GPDF command with the corresponding *SecurityLevel* and *SecurityKey* parameters from the Proxy Table. If the *SecurityLevel* and the *SecurityKey* do match, the Basic Proxy performs freshness check (see sec. A.3.6.1.2.1). If any of those checks fails and on reception of GP-DATA.indication with the Status AUTH_FAILURE or UNPROCESSED, the Basic Proxy stops processing the frame. The Basic Proxy SHALL NOT send GP Notification or GP Pairing Search.

If all the checks succeed, the Basic Proxy stores the *Sequence Number / Frame Counter* in the *GPD security frame counter* parameter of the Proxy Table entry for this GPD ID (and *Endpoint* matching or 0x00 or 0xff, if *ApplicationID* = 0b010), and constructs from the received GPDF a GP Notification command(s) for each group address and each unicast sink stored in the Proxy Table for this GPD; if *ApplicationID* = 0b010, the *Endpoint* field of the GP Notification command SHALL be set to the value of the *Endpoint* field from the triggering GPDF. The *BidirectionalCommunicationCapability* sub-field SHALL be set according to device capabilities; and the *gpTxQueueFull* sub-field of the *Options* field SHALL be set according to the status of this Basic Proxy's *gpTxQueue*, if the proxy does not support bidirectional communication, it SHALL set the *gpTxQueueFull* sub-field of the *Options* field to 0b1. The GPD *CommandID* and *GPD Command payload* are included in the clear in the GP Notification command, even if they were encrypted in the GPDF (*SecurityLevel* = 0b11, if supported); the MIC field from the GPDF SHALL NOT be included. The lower layers of the Basic Proxy stack (APS and NWK layer of Zigbee) will take care of appropriate protection of the command during tunneling through the Zigbee network. The *Ack. request* sub-field of the APS *Frame Control* field is set to 0b0.

If the proxy is not capable of bidirectional communication or if the *RxAfterTx* sub-field of the *Extended NWK Control Field* of the triggering GPDF was set to 0b0, for groupcast GP Notification, the Basic Proxy SHALL further use **proxy aliasing**, i.e. the following values: NWK Src address = alias source address (see A.3.6.3.3), NWK Sequence Number = alias sequence number (see A.3.6.3.3), NWK Dest address: 0xFFFD (broadcast to RxOnWhenIdle=TRUE); APS group address: as stored in the Proxy Table, APS source endpoint: Green Power EndPoint; APS counter = alias sequence number (see A.3.6.3.3),. The Basic Proxy SHALL send it after Dmin_u if *RxAfterTx* = 0b0, or else after *gppTunnelingDelay*, and SHALL NOT drop its own transmission upon reception of the same GP Notification.

For lightweight unicast GP Notification, the Basic Proxy SHALL NOT use **proxy aliasing**, i.e. NWK Src address, NWK sequence number and APS counter: are proxy's own values, NWK Dst address: sink's short address, APS source and destination end point: Green Power EndPoint. The Basic Proxy SHALL send it after Dmin.

A.3.5.2.4 Operation of sink side of GP Combo Basic

According to the current version of the specification, sinks joining a Zigbee SHALL set the *Involve TC* sub-field of the *gpsSecurityLevel* attribute as described in sec. A.3.3.2.6.

On receipt of GP Pairing Configuration command, the Basic Combo SHALL act as described in section A.3.5.2.4.1.

While in commissioning mode, the Basic Combo SHALL behave as described in sec. A.3.9.1, according to the supported commissioning functionality.

[164]In addition to the Device_annce sent as a result of successful proximity or multi-hop commissioning (see sec. A.3.9.1), a sink MAY also send Device_annce at other times, e.g. to prevent/resolve conflicts with devices not present at the time of the original announcement.

On receipt of GPD Decommissioning command, both in operational and in commissioning mode, the sink checks if it has a Sink Table entry for this GPD ID (and *Endpoint*, matching or 0x00 or 0xff, if *ApplicationID* = 0b010). If not, the frame is ignored. If yes, the sink decrypts the frame, if directly received, performs a freshness check, as described in A.3.6.1.2.1 and compares the *SecurityLevel* and *SecurityKeyType* with the values stored in the Sink Table entry. If any of those checks fails, the frame is silently dropped. If all those checks succeed, the sink removes this Sink Table entry, removes/replaces with generic entries the corresponding Translation Table entries if Translation Table functionality is supported, and removes Green Power EndPoint membership at APS level in the groups listed in the removed entry, if any. Then, the sink schedules sending of a GP Pairing command for this GPD ID (and *Endpoint*, matching or 0x00 or 0xff, if *ApplicationID* = 0b010), with the *RemoveGPD* sub-field set to 0b1. If the removed Sink Table entry included any pre-commissioned groups, and if the GPD Decommissioning command was received in commissioning mode, the sink SHALL also send GP Pairing Configuration message, with *Action* sub-field of the *Actions* field set to 0b100, *SendGPPairing* sub-field of the *Actions* field set to 0b0, and *Number of paired endpoints* field set to 0xfe.

On receipt of GP-SEC.request, the Combo Basic acts as described in sec. A.3.7.3.1.1.

On receipt of a GPD data command in operational mode via GP-DATA.indication with Status NO_SECURITY / SECURITY_SUCCESS or in GP Notification command, the sink checks the GPDID value: if SrcID = 0x00000000 (if *ApplicationID* = 0b000) or GPD IEEE address 0x0000000000000000 (if *ApplicationID* = 0b010), the frame SHALL be silently dropped.

[164] CCB #2408; resolution modified in 15-02014-013 as a result of Kavi comment #1378 from letter ballot for GP Bsic errata set: https://workspace.zigbee.org/higherlogic/ws/groups/PRO_GP/comments/view_comment?comment_id=1378

Then, the Basic Combo performs duplicate filtering, as described in A.3.6.1.2. Then the Basic Combo checks if it has a Sink Table entry for this GPD (and *Endpoint,* matching or 0x00 or 0xff, if *ApplicationID* = 0b010). If the Basic Combo does not have a Sink Table entry for this GPD (and *Endpoint,* matching or 0x00 or 0xff, if *ApplicationID* = 0b010), or the Sink Table entry exists but with another communication mode, and the incoming GP Notification message was received as lightweight or full unicast, the sink SHALL drop the command; it SHOULD broadcast a GP Pairing command for this GPD with the *AddSink* flag set to 0b1 and the correct value in the *CommunicationMode* sub-field and then a GP Pairing command for this GPD, the *CommunicationMode* flag set to the incorrect communication mode as in the triggering GP Notification, and *AddSink* flag set to 0b0. If the GPD command was received directly or in groupcast and the sink does not have a Sink Table entry for this GPD (and *Endpoint*, matching or 0x00 or 0xff, if *ApplicationID* = 0b010) and communication mode, the sink SHALL silently ignore it.
If the Basic Combo has a Sink Table entry for this GPD (and *Endpoint,* matching or 0x00 or 0xff, if *ApplicationID* = 0b010), and if the received GPD command is a Data command (0x00 <= CommandID <= 0xDF), the value of the sub-fields *SecurityLevel* and *SecurityKey* from the received command are compared with the corresponding *SecurityLevel* and *SecurityKeyType* parameters from the Sink Table. If the *SecurityLevel* and the *SecurityKey* do match, the Basic Combo performs a freshness check, as described in A.3.6.1.2.1. If any of those checks fails, the frame is silently dropped. If all those checks succeed, the Basic Combo updates the *GPD security frame counter* parameter of this Sink Table entry.
If all previous checks succeed, the Combo Basic SHALL accept the GPD commands received in GP Notification with *ProxyInfoPresent* sub-field of the *Options* field set to 0b0. Then if the Basic Combo has a Translation Table, the Basic Combo checks the value of the *EndPoint* field of the Translation Table entries for the GPD. If there is a Translation Table with value of the *EndPoint* field other than 0x00 and 0xfd, the Basic Combo SHALL also translate the GPD command into a Zigbee command, as indicated in the Translation Table entry, and send it to the paired local endpoint(s), as indicated in the *EndPoint* field, for execution.

If the Basic Combo has a Sink Table entry for this GPD (and *Endpoint,* matching or 0x00 or 0xff, if *ApplicationID* = 0b010), the Basic Combo is in operational mode and if the received GPD command is either a GPD Commissioning command, the Basic Combo SHALL NOT enter commissioning mode and SHALL NOT perform any commissioning action. The Basic Combo MAY provide some indication to the user about the attempted commissioning action. Other GPD commissioning commands received in operational mode SHALL be silently dropped, unless their handling in operation is explicitly described.

The Combo Basic device SHALL act upon a GPD command from a paired GPD just once and SHALL filter out duplicate GPD commands received in both direct and tunneled mode (i.e. via both client and server side of the Green Power cluster).
On receiving a GPD frame in direct mode, the GP Combo Basic device SHALL NOT only forward it to local paired end points, but also participate in forwarding this frame to other sinks listed in its Proxy Table for this GPD (and *GPD Endpoint*, matching or 0x00 or 0xff, if *ApplicationID* = 0b010), if any, as specified in section A.3.5.2.1.
The proxy side of the combo SHALL create a Proxy Table entry for a GP Pairing using Pre-commissioning groupcast if it is sent by the sink side residing on the same radio. Since a broadcast transmission is typically not passed up again to the originating endpoint, this may require special solution in the combo code. The proxy side of the combo is not required to create a Proxy Table entry for a GP Pairing using DGroup or unicast communication mode if it is sent by the sink side residing on the same radio.
The proxy side of the combo is not required to enter the commissioning mode for a GP Proxy

Commissioning Mode with *Action* = *Enter* if it is sent by the sink side residing on the same radio.

Green Power cluster commands related to the GP functionality not supported by the Basic Combo (see sec. A.3.2.9 - A.3.2.10) SHALL be silently dropped.

The SDL diagram illustrating the Basic Proxy behavior in operational and commissioning mode is included in sec. A.3.8.1.

A.3.5.2.4.1 Handling of GP Pairing Configuration

The sink's reaction on reception of GP Pairing Configuration command (see sec. A.3.3.4.6) is the same, irrespective of whether it is in commissioning mode or operational mode.

On receipt of GP Pairing Configuration command, the sink is requested to update its Sink Table and Translation Table, if supported, based on the value of the *Action* sub-field of the *Actions* field and using the data provided in the remaining fields, as follows.

A received GP Pairing Configuration command carrying SrcID = 0x00000000 (if *ApplicationID* = 0b000) or GPD IEEE address 0x0000000000000000 (if *ApplicationID* = 0b010) SHALL be silently dropped; Sink Table entry SHALL NOT be created or updated. GP Pairing Configuration command with SrcID = 0xffffffff (if *ApplicationID* = 0b000) or GPD IEEE address 0xffffffffffffffff (if *ApplicationID* = 0b010) denotes a pairing for all GPD with a particular *ApplicationID* and SHALL be created if there is space in the Sink Table.

[165]If the *GPD ID* field of a received GP Pairing Configuration command carries SrcID from the valid range 0x00000001 – 0xfffffff8 (if *ApplicationID* = 0b000) or GPD IEEE address from the valid range (if *ApplicationID* = 0b010), the sink SHALL proceed as follows.

If the *Action* sub-field of the *Actions* field was set to 0b000, 0b001 or 0b010 and the *SecurityLevel* field of the *SecurityUse* field is set to 0b01, the sink SHALL NOT update (if existent) nor create a Sink Table entry for this GPD ID (and *Endpoint*, matching or 0x00 or 0xff, if *ApplicationID* = 0b010). If the command was sent in unicast, it MAY send ZCL Default Response Command with the *Status* code field indicating FAILURE (see [3]).

If the *Action* sub-field of the *Actions* field is set to 0b000, the sink SHALL NOT modify the Sink Table nor the Translation Table. If the *Send GP Pairing* sub-field of the *Actions* field of the GP Pairing Configuration command is set to 0b1, and there is an entry for this GPD ID (and *Endpoint*, matching or 0x00 or 0xff, if *ApplicationID* = 0b010) in the Sink Table, the sink SHALL send the GP Pairing command with *AddSink* = 0b1 and *RemoveGPD* = 0b0 for all information available in the Sink Table entry. If the *Send GP Pairing* sub-field of the *Actions* field of the GP Pairing Configuration command is set to 0b1, but there is no entry for this GPD ID (and *Endpoint*, matching or 0x00 or 0xff, if *ApplicationID* = 0b010) in the Sink Table, the sink SHALL NOT send the GP Pairing command(s).

Action sub-field equal to 0b001 or 0b010

[165] CCB #2360; Resolution added in 15-02014-011

For *Action* sub-field equal to 0b001 or 0b010, the sink starts as follows. The sink checks if it supports the *SecurityLevel* requested (i.e., if it is higher than [166]or equal to the *gpsSecurityLevel*) and if it supports the requested *CommunicationMode* (as indicated in the *gpsFunctionality/gpsActiveFunctionality* attribute). If either of those checks fails, it drops the frame; Sink Table and Translation Table is not modified. If the command was sent in unicast, it MAY send ZCL Default Response Command with the *Status* code field indicating FAILURE (see [3]). If both checks succeed, the sink proceeds as follows, depending on the *Action* sub-field value. [167]If the *GPD Application Description command follows* sub-field of the *Application Information* field is set to 0b1, the sink SHALL buffer the received information in an application-specific manner and SHALL start the *MultiSensorCommissioningTimeout* timer, if not running yet.

[168]If the *GPD Application Description command follows* sub-field of the *Application Information* field is set to 0b0 OR if the *GPD Application Description command follows* sub-field of the *Application Information* field is set to 0b1 and the complete commissioning information consisting of GP Pairing Configuration command for this GPD with *Action* = 0b001 or 0b010 (add or replace) and all the Report Descriptors (as can be derived from the fields *Total number of reports*) for a GPD were received, the sink proceeds as follows.

[166] CCB #1978; Resolution added in 15-02014-002
[167] Comment #777 from GP multi-sensor v0.7 letter ballot, GP multi-sensor LB v0.9 comment #973: https://workspace.zigbee.org/kws/groups/PRO_GP/comments/view_comment?comment_id=973
[168] Comment #777 from GP multi-sensor v0.7 letter ballot, GP multi-sensor LB v0.9 comment #973: https://workspace.zigbee.org/kws/groups/PRO_GP/comments/view_comment?comment_id=973

If the *Action* sub-field of the *Actions* field is set to 0b010, the sink SHALL remove all the Sink Table entry/entries for this GPD (and *Endpoint*, matching or 0x00 or 0xff, if *ApplicationID* = 0b010), if any. For all the removed groupcast pairings, the sink SHALL remove its Green Power EndPoint as a member of the group at APS level. If the sink has any Translation Table entry/entries for this specific GPD ID (and *GPD Endpoint*, matching or 0x00 or 0xff, if *ApplicationID* = 0b010), they all SHALL be removed or replaced with the generic Translation Table entry. Both for *Action* sub-field equal to 0b001 if there is no Sink Table entry for this GPD ID (and *Endpoint*, matching or 0x00 or 0xff, if *ApplicationID* = 0b010) and 0b010, the sink SHALL then analyze the *Number of paired endpoints* field.
If the *Number of paired endpoints* field is set to 0x00 or 0xfd, the data from this GPD is not meant for local execution on this sink. If the sink does support *Sink Table-based forwarding* in the requested CommunicationMode, it SHALL create a Sink Table entry with the supplied information and a Translation Table entry for the GPD ID (and *GPD Endpoint*, matching or 0x00 or 0xff, if *ApplicationID* = 0b010), with the *EndPoint* field having the value 0xfd. If the *CommunicationMode* supplied in the Pairing Configuration command was groupcast, the sink SHALL add its Green Power EndPoint as a member of the supplied group or derived group at APS level if not already a member. If the sink does NOT support *Sink Table-based forwarding* or it does not support *Sink Table-based forwarding* in the requested *CommunicationMode*, the sink (i) MAY create a Sink Table entry with the supplied information and a Translation Table entry for this GPD ID (and *GPD Endpoint*, matching or 0x00 or 0xff, if *ApplicationID* = 0b010) with *Endpoint* field set to 0x00; (ii) MAY create a Sink Table entry with the supplied information and refrain from creating any Translation Table entry for this GPD ID (and matching *GPD Endpoint*, if *ApplicationID* = 0b010) (sink SHALL NOT use this option if it has generic Translation Table entries for this GPD command(s)); or (iii) MAY refrain from creating both Sink Table entry and Translation Table entry for this GPD ID (and matching *GPD Endpoint*, if *ApplicationID* = 0b010). If the Sink Table entry is created and the *CommunicationMode* supplied in the Pairing Configuration command was groupcast, the sink SHALL add its Green Power EndPoint as a member of the supplied group or derived group at APS level if not already a member.
If the *Number of paired endpoints* field is set to 0xff, all matching endpoints are to be paired; the sink MAY then create a Sink Table entry with the supplied information and Translation Table entry for the GPD ID (and *GPD Endpoint*, matching or 0x00 or 0xff, if *ApplicationID* = 0b010), with the *EndPoint* field having the value 0xff; the unmodified generic entry, if available, MAY be used instead. If the *CommunicationMode* supplied in the Pairing Configuration command was groupcast, the sink SHALL add its Green Power EndPoint as a member of the supplied group or derived group at APS level if not already a member. If no match is found, the sink SHALL act as described above for *Number of paired endpoints* equal to 0x00 or 0xfd.
If the *Number of paired endpoints* field is set to 0xfe, the paired endpoints are to be derived by the sink. If the GP Pairing Configuration command carries a *CommunicationMode* 0b10 and the *GroupList* is present, all application endpoints being members of this group are to be paired; otherwise, the sink is to derive the paired endpoints in an application-specific manner. The sink SHOULD then create a Sink Table entry with the supplied information and Translation Table entry/entries for the GPD ID (and *GPD Endpoint*, matching or 0x00 or 0xff, if *ApplicationID* = 0b010), with the *EndPoint* field containing the derived value of the sink's endpoint; the unmodified generic entry, if available, MAY be used instead. If the *CommunicationMode* supplied in the Pairing Configuration command was groupcast, the sink SHALL add its Green Power EndPoint as a member of the supplied group or derived group at APS level if not already a member. If no match is found [169](i.e., in case of *CommunicationMode* 0b10, none of the application endpoints of the sink is a member of any of the groups listed in the *GroupList* field), the sink SHALL act as described above for *Number of paired endpoints* equal to 0x00 or 0xfd.
If the *Number of paired endpoints* field has values other than 0x00, 0xfd, 0xfe, or 0xff, the *Paired end-*

[169] CCB #2169; Resolution added in 15-02014-005

points field is present and contains the list of local endpoints paired to this GPD; the sink creates a Translation Table entry for this GPD ID (and *GPD Endpoint*, if *ApplicationID* = 0b010) and each EndPoint listed in the *Paired endpoints* field. If the *CommunicationMode* supplied in the Pairing Configuration command was groupcast, the sink SHALL add its Green Power EndPoint as a member of the supplied group or derived group at APS level if not already a member.

If the *Action* sub-field of the *Actions* field is set to 0b001 and a Sink Table entry for this GPD (and *Endpoint*, matching or 0x00 or 0xff, if *ApplicationID* = 0b010) already exists, the sink checks the match between the *CommunicationMode* in the GP Pairing Configuration command and the Sink Table entry. If the existing entry contains different *CommunicationMode*, the existing entry SHALL NOT be overwritten; new entry MAY be created, storing the supplied information; if the supplied information is not stored and if the command was sent in unicast, the sink MAY send ZCL Default Response Command with the *Status* code field indicating FAILURE (see [3]). If the *CommunicationMode* does match, the sink checks the *Number of paired endpoints* field. If set to 0xff, 0xfe or value other than 0x00 or 0xfd; the sink SHALL attempt extending the Sink Table and/or Translation Table entry with the supplied information (if not already listed there). If the Sink Table entry is updated and the *CommunicationMode* supplied in the Pairing Configuration command was groupcast, the sink SHALL add its Green Power EndPoint as a member of the supplied group or derived group at APS level if not already a member.

Action sub-field equal to 0b101

[170]If the *Action* sub-field of the *Actions* field is set to 0b101, if the *MultiSensorCommissioningTimeout* is not running, the sink SHALL start it; if it is running, the sink SHALL NOT modify it. Then, the sink SHALL analyze the supplied *Report Descriptor* fields; in case of application functionality match. If there is application functionality match AND [171]the sink received GP Pairing Configuration command for this GPD with *Action* = 0b001 or 0b010 (add or replace) AND the sink received all Report Descriptors for this GPD (as can be derived from the fields *Total number of reports*), then the sink SHALL complete the pairing procedure by updating the Sink Table entry as triggered by the GP Pairing Configuration command for this GPD with *Action* = 0b001 or 0b010 (add or replace), as described above, and by storing the information about the matching Data Point Descriptors – if the Translation Table functionality is supported, then in the *Additional information block* field of the Translation Table entry for that *SrcID/GPD IEEE address* (and *Endpoint*, matching or 0x00 or 0xff, if *ApplicationID* = 0b010), and if the Translation Table functionality is not supported, in an application-specific way.

[172]To increase the robustness of the commissioning process, the sink SHALL be capable of receiving the GP Pairing Configuration commands with *Action* sub-field of the *Actions* field is set to 0b101 carrying Application Description GPDFs out of order and in duplicate.

[173]If the sink did NOT receive GP Pairing Configuration command for this GPD with *Action* = 0b001 or 0b010 (add or replace) OR all the Report Descriptors (as can be derived from the fields *Total number of reports*) for a GPD, the sink SHALL buffer the information received in an application-specific manner and continue waiting until *MultiSensorCommissioningTimeout*.

[170] GP multi-sensor LB v0.9 comment #973: https://workspace.zigbee.org/kws/groups/PRO_GP/comments/view_comment?comment_id=973
[171] Comment #5 from GP multi-sensor August PoC, Zigbee document 16-02611
[172] Comment #777 from GP multi-sensor v0.7 letter ballot, GP multi-sensor LB v0.9 comment #973: https://workspace.zigbee.org/kws/groups/PRO_GP/comments/view_comment?comment_id=973
[173] Comment #777 from GP multi-sensor v0.7 letter ballot, GP multi-sensor LB v0.9 comment #973: https://workspace.zigbee.org/kws/groups/PRO_GP/comments/view_comment?comment_id=973

Upon *MultiSensorCommissioningTimeout*, if the sink did NOT receive GP Pairing Configuration command for this GPD with *Action* = 0b001 or 0b010 (add or replace) OR all the Report Descriptors (as can be derived from the fields *Total number of reports*) for a GPD, the sink SHALL drop all the buffered information and SHALL NOT create any Sink Table or Translation Table entries for this GPD.

Action sub-field equal to 0b011 or 0b100

If the *Action* sub-field of the *Actions* field is set to 0b011, the sink SHALL check if it has Sink Table entry for the supplied *SrcID/GPD IEEE address* (and *Endpoint*, matching or 0x00 or 0xff, if *ApplicationID* = 0b010) with the supplied *CommunicationMode* and, in case of groupcast *CommunicationMode*, the supplied GroupID. If yes, this pairing SHALL be removed. In case of groupcast, the sink SHALL remove its Green Power EndPoint as a member of this group at APS level. If the sink has any Translation Table entry/entries for this GPD ID (and *GPD Endpoint*, matching or 0x00 or 0xff, if *ApplicationID* = 0b010) and sink's endpoint, if specific endpoint is provided in the GP Pairing Configuration command, they SHALL be removed/replaced with the generic Translation Table entry.

If the *Action* sub-field of the *Actions* field is set to 0b100, the sink SHALL remove all the Sink Table entry(s) for this GPD and *Endpoint*, matching or 0x00 or 0xff, if *ApplicationID* = 0b010, if they exist. For all the pairings that were for groupcast, the sink SHALL remove its Green Power EndPoint as a member of the group at APS level. If the sink has any Translation Table entry/entries for this GPD ID (and *GPD Endpoint*, if *ApplicationID* = 0b010), they all SHALL be removed/replaced with the generic Translation Table entry.

Action sub-field equal to 0b000 – 0b100

If the *Send GP Pairing* sub-field of the *Actions* field of the GP Pairing Configuration command is set to 0b1, the sink SHALL, upon completion of Sink Table update, send the GP Pairing command(s) reflecting the changes made, [174]i.e. if a pairing was added as a result of *Action* set to 0b001 or 0b010, the sink SHALL send the GP Pairing command with *AddSink* = 0b1 and *RemoveGPD* = 0b0 for all information available in the Sink Table entry; if a pairing was removed as a result of *Action* set to 0b011, the sink SHALL send the GP Pairing command with *AddSink* = 0b0 and *RemoveGPD* = 0b0; if a pairing was removed as a result of *Action* set to 0b100, the sink SHALL send the GP Pairing command with *AddSink* = 0b0 and *RemoveGPD* = 0b1. If a pairing was added, the sink SHALL send a Device_annce command for the alias (with the exception of lightweight unicast communication mode). If the *Send GP Pairing* sub-field of the *Actions* field was set to 0b0, the sink SHALL NOT send the GP Pairing command or Device_annce command.

A.3.5.2.5 Sink operation

On receipt of GP Pairing Configuration command, a sink SHALL act as described in section A.3.5.2.4.1.

A sink SHOULD re-announce its pairings when it rejoins the network (e.g. after being powered off) by sending a GP Pairing command.

[174] CCB #2323; Resolution added in 15-02014-011

On receipt of Zigbee Update Device and Device_annce commands with IEEE address other than 0xffffffffffffffff, the sink SHALL check if the NWKAddr field matches any of the aliases used by this sink. If that's the case, an address conflict is with a regular Zigbee device is discovered and the sink SHALL act according to Zigbee [1] address conflict announcement procedure, i.e. the proxy SHALL send after randomly chosen delay from between Dmin and Dmax (see A.3.6.3.1) the Zigbee Device_annce command (unless identical frame was received within this time), formatted as described inA.3.6.3.4.2, using the conflicting Alias NWK source address , to force the regular Zigbee device to change its short address. The alias SHALL NOT be changed.

On receipt in operational mode of a GP Notification carrying GPD Commissioning command for a GPD the sink has Sink Table entry for, the sink SHALL silently drop the frame; the sink SHALL NOT open commissioning mode. If the security check was successful, the sink MAY perform other actions, e.g. indicate the attempted (de-)commissioning to the user.

On receipt of GP-SEC.request, the sink acts as described in sec. A.3.7.3.1.1.

On receipt of a GP Commissioning Notification with *SecurityProcessingFailed* sub-field of the *Options* field set to 0b0, the sink performs duplicate filtering, as described in A.3.6.1.2. Then, and on receipt of GP-DATA.indication with the Status SECURITY_SUCCESS for the GPD Decommissioning command, GPD Commissioning command and GPD Data command with *Auto-Commissioning* sub-field set to 0b1, if supported, the sink checks if it is in commissioning mode. If not, the GP Commissioning Notification command, and Commissioning GPDF is silently dropped; the sink SHALL NOT open commissioning mode. The sink MAY perform other actions, e.g. indicate the attempted (de-)commissioning to the user.

On receipt of GPD Decommissioning command, the sink checks if it has a Sink Table entry for this GPD (and *Endpoint*, matching or 0x00 or 0xff, if *ApplicationID* = 0b010). If not, the frame is ignored. If yes, the sink performs a freshness check, as described in A.3.6.1.2.1 and compares the SecurityLevel and SecurityKeyType with the values stored in the Sink Table entry. If any of those checks fails, the frame is silently dropped. If all those checks succeed, the sink removes this Sink Table entry, removes/replaces with generic entries the corresponding Translation Table entries if Translation Table functionality is supported, and removes Green Power EndPoint membership at APS level in the groups listed in the removed entry, if any. Then, the sink schedules sending of a GP Pairing command for this GPD (and *Endpoint*, matching or 0x00 or 0xff, if *ApplicationID* = 0b010), with the *RemoveGPD* sub-field set. If the removed Sink Table entry included any pre-commissioned groups, and if the GPD Decommissioning command was received in commissioning mode, the sink SHALL send GP Pairing Configuration message, with *Action* sub-field of the *Actions* field set to 0b100, *SendGPPairing* sub-field of the *Actions* field set to 0b0, and *Number of paired endpoints* field set to 0xfe.

If the sink supports proximity commissioning or Multi-hop commissioning functionality is in commissioning mode and the GPDF was a Commissioning GPDF or a Data GPDF with *Auto-Commissioning* sub-field set to 0b1, the sink behaves as described in sec. A.3.9.1.

On receipt of a GP Proxy Commissioning Mode command or a GP Tunneling Stop command, the sink silently drops those commands, irrespective of whether it is in operational mode or in commissioning mode.

If the sink implements the Proxy table maintenance functionality, the sink SHALL act as follows. The sink's reaction on reception of GP Pairing Search is the same, irrespective of whether it is in commissioning mode or operational mode.

On receipt of a GP Pairing Search command, a sink checks if it has a Sink Table entry for this GPD (and *Endpoint*, matching or 0x00 or 0xff, if *ApplicationID* = 0b010) and the communication mode requested by the flags *RequestUnicastSinks*, *RequestDerivedGroupcastSinks*, and *RequestCommissionedGroupcastSinks* in the *Options* field of the received GP Pairing Search command. If not, the command is ignored. If yes, the sink sends a GP Pairing command with the *Options* field set as follows: *AddSink* set to 0b1, *RemoveGPD* set to 0b0, *CommunicationMode* and *GPDfixed* corresponding to the values in the *Options* parameter of the Sink Table entry, *SecurityLevel and SecurityKeyType* corresponding to the values in the *Security Options* parameter of the Sink Table entry. It includes the fields *GPD Security Frame Counter* and *GPD Security Key*, if they were requested by the flags *Request GPD Security Frame Counter* or *Request GPD Security key* in the *Options* field of the received GP Pairing Search command being set to 0b1. On receipt of a broadcast GP Notification, a sink checks if it has a Sink Table entry for this GPD (and *Endpoint*, matching or 0x00 or 0xff, if *ApplicationID* = 0b010). If the *SecurityLevel* and *SecurityKeyType* check, freshness check and security processing all pass successfully, the sink executes the command, and then sends GP Pairing command, with the values in the *Options* field reflecting the requested communication mode options and the required fields present (at the minimum the *GPD security frame counter*). If the sink sends the GP Pairing command with *AddSink* sub-field set to 0b1, it SHALL also send Device_annce for the corresponding alias (with the exception of lightweight unicast communication mode).

On reception of GP-DATA.indication with Status AUTH_FAILURE, the sink SHALL silently drop it.

On receipt of a GPD data command in operational mode, either in tunneled mode via GP Notification command or in via GP-DATA.indication, with Status NO_SECURITY / SECURITY_SUCCESS, if the sink has GP stub implemented, the sink performs duplicate filtering, as described in A.3.6.1.2. Then the sink checks if it has a Sink Table entry for this GPD (and *Endpoint*, matching or 0x00 or 0xff, if *ApplicationID* = 0b010). If not, and the GPD command was received in unicast GP Notification, and the sink supports full unicast communication, it schedules sending of GP Notification Response, if supported, in unicast to the originating proxy, with the GPD ID and, if *ApplicationID* = 0b010, *Endpoint* field copied from the incoming GP Notification message, the *No Pairing* sub-field set to 0b1, as well as broadcasting of a GP Pairing command with the *CommunicationMode* flag set to the lightweight or full unicast communication mode, as used by this sink (0b11 or 0b00) and *AddSink* flag set to 0b0. If the sink does not have a Sink Table entry for this GPD (and *Endpoint*, matching or 0x00 or 0xff, if *ApplicationID* = 0b010), and the GPD command was received directly or in groupcast, the command is silently ignored. If the sink has a Sink Table entry for this GPD (and *Endpoint*, matching or 0x00 or 0xff, if *ApplicationID* = 0b010) for groupcast communication mode (0b01 or 0b10) and it receives unicast GP Notification, the sink SHALL send GP Notification Response, if supported, unicast to the originating proxy, with the *No Pairing* flag set to 0b1 and *First to Forward* set according to the duplicate filter status; and SHOULD broadcast a GP Pairing command, whereby the destination endpoint is set to 0xf2, with the *AddSink* flag set to 0b1 and the correct groupcast value in the *CommunicationMode* sub-field; and then GP Pairing command with GPD ID and, if *ApplicationID* = 0b010, *Endpoint* field copied from corresponding Sink Table entry, the *CommunicationMode* flag set to the lightweight or full unicast communication mode, as used by this sink (0b11 or 0b00) and *AddSink* flag set to 0b0.

If the sink does have a Sink Table entry for this GPD (and *Endpoint*, matching or 0x00 or 0xff, if *ApplicationID* = 0b010), and the communication mode was correct, the value of the sub-fields *SecurityLevel* and *SecurityKey* from the received command are compared with the corresponding *SecurityLevel* and *SecurityKeyType* parameters from the Sink Table. If the *SecurityLevel* and the *SecurityKey* do match, and for GP-DATA.indication, the sink performs a freshness check, as described in A.3.6.1.2.1. If any of those checks fails, the frame is silently dropped. If all those checks succeed, the sink updates the *GPD security frame counter* parameter of this Sink Table entry, if present, and proceeds as follows. If all previous checks succeed, the sink SHALL accept GPD commands received in GP Notification with *ProxyInfoPresent* sub-field of the *Options* field set to 0b0.

If the sink supports the *Sink Table-based groupcast forwarding* functionality, and the GPD command was received directly in GP-DATA.indication, and the Sink Table entry for the GPD (and *Endpoint*, matching or 0x00 or 0xff, if *ApplicationID* = 0b010) indicates any groupcast *CommunicationMode*, and there is no Translation Table (if supported) entry for this GPD ID (and *GPD Endpoint*, matching or 0x00 or 0xff, if *ApplicationID* = 0b010) and GPD CommandID with *endpoint* field set to 0x00, the sink SHALL construct and send a GP Notification command for each of the paired groups, taking the following parameters from the Sink Table entry: *CommunicationMode* subfield of the *Options* field; *GroupList* field if present or otherwise derived groupcast; *AssignedAlias* field if present or otherwise derived alias; *Radius* field if present or otherwise default radius; and security settings, if present. The *BidirectionalCommunicationCapability* sub-field SHALL be set according to device capabilities, and the *gpTxQueueFull* sub-field of the *Options* field SHALL be set according to the status of this sink's *gpTxQueue* (i.e., if there is no entry in the *gpTxQueue* for this GPD and the queue is full, it sets the *gpTxQueueFull* sub-field to 0b1, otherwise if it has an entry for this GPD or at least one empty entry, it sets it to 0b0); if the sink does not support bidirectional communication, it SHALL set the *gpTxQueueFull* sub-field of the *Options* field to 0b1.

Then, the sink checks if the command requires response. If the received GPD command does not require response, the sink executes the command. To do this, if the sink has a Translation Table, the sink checks the value of the *EndPoint* field of the Translation Table entries for the GPD. If there is a Translation Table, generic or dedicated, with value of the *EndPoint* field other than 0x00 and 0xfd, the sink SHALL also translate the GPD command into a Zigbee command, as indicated in the Translation Table entry, and send it to the paired local endpoint(s), as indicated in the *EndPoint* field, for execution.

If the received GPD command requires response, and the sink supports bidirectional communication, the sink checks if the GPD requesting it is capable of bidirectional communication in operation. This information is available in the *RxOnCapability* sub-field of the *Options* field of the Sink Table entry for this GPD. If yes, the sink selects TempMaster as described in sec. A.3.6.2.3. If the sink itself is selected as TempMaster, the sink calls GP-DATA.request, with the required *GPD CommandID* and *GPD Command Payload*.

The sink behavior in the following situations will be defined by the application profile: (i) on receipt of Data GPDF in commissioning mode, (ii) on receipt of a GP Commissioning Notification with *SecurityProcessingFailed* sub-field of the *Options* field set to 0b1. Also for situations covered in this section, application profiles MAY define additional actions.

In sec. A.3.7.3.2, SDL diagrams for the above described operation are provided.

A.3.5.2.6 GP Combo operation

If the device is a GP Combo device, i.e. has the functionality of both the proxy and the GPT+, it SHALL perform all the actions specified in sections A.3.5.2.1 and A.3.5.2.4.

Specifically, the Combo device SHALL act upon a GPD command from a paired GPD just once and SHALL filter out duplicate GPD commands received in both direct and tunneled mode (i.e. via both client and server side of the Green Power cluster).
On receiving a GPD frame in direct mode, the GP Combo device SHALL NOT only forward it to local paired end points, but also participate in forwarding this frame to other sinks listed in its Proxy Table for this GPD (and *GPD Endpoint*, matching or 0x00 or 0xff, if *ApplicationID* = 0b010), if any, as specified in section A.3.5.2.1.
The proxy side of the combo SHALL create a Proxy Table entry for a GP Pairing using Pre-commissioning groupcast if it is sent by the sink side residing on the same radio. Since a broadcast transmission is typically not passed up again to the originating endpoint, this may require special solution in the combo code. The proxy side of the combo is not required to create a Proxy Table entry for a GP Pairing using DGroup or unicast communication mode if it is sent by the sink side residing on the same radio.
The proxy side of the combo is not required to enter the commissioning mode for a GP Proxy Commissioning Mode with *Action* = *Enter* if it is sent by the sink side residing on the same radio.

A.3.6 GP Implementation details

A.3.6.1 Generic

This chapter describes functionality common to all Green Power cluster implementations, both on proxies and sinks.

A.3.6.1.1 Broadcast

Whenever NWK level broadcast transmission is mentioned within this specification without further description for the GP-defined commands, or where no further description is provided by the Zigbee specification by the Zigbee-defined commands, the RxOnWhenIdle=TRUE (0xfffd) broadcast address SHALL be used.

Whenever broadcast communication without APS-level multicast aka groupcast is used for transporting Green Power cluster messages, the destination endpoint SHALL be set to 0xf2.

A.3.6.1.2 Duplicate filtering

In the Green Power EndPoint duplicate filter, each entry is stored for a finite time of *gpDuplicateTimeout* and is used to filter both direct and tunneled GPD commands.

If the GPD command used *SecurityLevel* 0b00, the filtering of duplicate GPD messages is based on the *MAC sequence number* of a particular GPD, identified by GPD ID. If the GPD command used *SecurityLevel* 0b10 or 0b11, then the filtering of duplicate messages is performed based on the *GPD security frame counter*.

If the receiving device is:

- a proxy,
- a sink and it does not support bidirectional communication,
- a sink does support the bidirectional communication but the *RxAfterTx* sub-field is set to 0b0,

of all instances of any GPD command received – both directly as GPDF or indirectly in a GP command - only one instance, received in the correct communication mode, SHALL be processed.

If the device is a sink, it does support the bidirectional communication and the *RxAfterTx* sub-field is set to 0b1, then the sink processes further - independent of the manner of receiving the GPD command: directly as GPDF or indirectly in a GP command - each further instance of this command with *BidirectionalCommunicationCapability* = 0b1 and either with *GPP-GPD link* better than the last received one (whereby better *GPP-GPD link* is defined as one having higher value of the *Link quality* sub-field, and if *Link quality* is equal, as one having higher value of the *RSSI* sub-field), or by the same *GPP-GPD link* – with the lower short address. The *GPP-GPD link* value and the address SHALL then be also stored.

In case of duplicate full unicast GP Notification, the sink SHALL send GP Notification Response, if supported, unicast to the originating proxy (information available from NWK header of the received GP Notification) with the *FirstToForward* flag is set to 0b0. The duplicate groupcast/broadcast GP Notifications are dropped silently.

Table 47 summarizes the duplicate filtering in the sink's Green Power EndPoint, dependent on the required and received *CommunicationMode* and the *RxAfterTx* value.

Table 47 – Duplicate filtering in the sink

Required communication mode	Communication mode of first packet	RxAfterTx (Appoint TempMaster)	Action
Derived group	Full/lightweight Unicast	TRUE/FALSE	Drop packet, don't store the new values in the duplicate filter, send GP Notification Response, if supported, unicast to the originating proxy, with the *FirstToForward* sub-field of the *Options* field set to 0b0; GP Pairing command with the *AddSink* flag set to 0b1 and the correct groupcast value in the *CommunicationMode* sub-field; and then GP Pairing command with the *CommunicationMode* flag set to 0b00 or 0b11, as supported, and *AddSink* flag set to 0b0.
Pre-commissioned group	Full/lightweight Unicast	TRUE/FALSE	
Full/lightweight Unicast, Pre-commissioned group	Derived group	TRUE/FALSE	drop packet, don't store the new values in the duplicate filter
Full/lightweight Unicast, Derived group	Pre-commissioned group	TRUE/FALSE	
Derived group	Derived group	FALSE	pass packet up, store the new values in the duplicate filter
Pre-commissioned group	Pre-commissioned group		
Any	GPDF (direct mode)	FALSE	pass packet up, store the new values in the duplicate filter
any	broadcast	FALSE	Recommended: pass packet up, store the new values in the duplicate filter, send GP Pairing with the proper communication mode; can be modified by the profile
Full Unicast	Full Unicast	FALSE	For the first received full unicast packet: Send GP Notification Response with *FirstToForward* sub-field of the *Options* field set to 0b1, pass packet up, store the new values in the duplicate filter For the subsequent received unicast packets: Send GP Notification Response with *FirstToForward* sub-field of the *Options* field set to 0b0 (even if retry from the *FirstToForward* proxy), drop packet
Derived group	Derived group	TRUE	pass packet up if *BidirectionalCommunicationCa-*

zigbee alliance

Required communication mode	Communication mode of first packet	RxAfterTx (Appoint TempMaster)	Action
Pre-commissioned group	Pre-commissioned group	TRUE	*pability* = 0b1 and better GPP-GPD link value (or same GPP-GPD link value, lower address), store the new values in the duplicate filter
Any	GPDF (direct mode)	TRUE	pass packet up if *BidirectionalCommunicationCapability* = 0b1 and better GPP-GPD link value (or same GPP-GPD link value, lower address), store the new values in the duplicate filter, send GP Pairing with the proper communication mode
Any	broadcast	TRUE	Recommended: pass packet up if *BidirectionalCommunicationCapability* = 0b1 and better GPP-GPD link value (or same GPP-GPD link value, lower address), store the new values in the duplicate filter, send GP Pairing with the proper communication mode; can be modified by the profile
Full Unicast	Full Unicast	TRUE	For the first received full unicast packet: Send GP Notification Response with *FirstToForward* sub-field of the *Options* field set to 0b1, pass packet up if better GPP-GPD link value (or same GPP-GPD link value, lower address), store the new values in the duplicate filter For the subsequent received full unicast packets: Send GP Notification Response with *FirstToForward* sub-field of the *Options* field set to 0b0 (even if retry from the *FirstToForward* proxy), pass packet up if *BidirectionalCommunicationCapability* = 0b1 and better GPP-GPD link value (or same GPP-GPD link value, lower address)
Lightweight unicast	Lightweight unicast	TRUE/FALSE	pass packet up, store the new values in the duplicate filter; subsequent packets MAY be passed up proxy selection, but SHALL NOT be executed multiple times

A.3.6.1.2.1 gpDuplicateTimeout

The time the Green Power EndPoint of the sink and the proxy keeps the information on the received GPDF, in order to filter out duplicates.

The default value of 2 seconds can be modified by the application profile.

A.3.6.1.3 Freshness check

If the GPD command used *SecurityLevel* 0b00, any number that passes the duplicate filter is accepted.

If the GPD command used *SecurityLevel* 0b10 or 0b11, then the filtering of duplicate messages is performed based on the *GPD security frame counter*, stored in the Proxy/Sink Table entry for this GPD (and *GPD Endpoint*, matching or 0x00 or 0xff, if *ApplicationID* = 0b010). The received *GPD security frame counter* must be higher than the value stored in the Proxy/Sink Table; roll over SHALL NOT be supported.

When a new incremental value is being accepted, the corresponding parameter of the Proxy/Sink Table entry SHALL be updated.

A.3.6.1.4 Derived groupcast (DGroupID)

Usage of the derived groupcast *CommunicationMode* allows for NWK/APS level filtering at the routers forwarding the tunneled message, as well as at the sinks.

The GroupID for the derived groupcast mode, DGroupID, SHALL be derived from the GPD ID in exactly the same way as the alias source address (see A.3.6.3.3).

If *ApplicationID* = 0b010, the GPD *Endpoint* SHALL NOT be included in the alias/DGroupID calculation.

A.3.6.1.5 Bidirectional communication

A.3.6.1.5.1 Payload sizes

The payload of any GPD command sent by the sink to the GPD SHALL NOT exceed:

- For a GPD with *ApplicationID* = 0b000: 64 octets;
- For a GPD with *ApplicationID* = 0b010: 59 octets.

This limitation is introduced to avoid fragmentation, or dropping the command, if fragmentation is not supported, in the case a remote device (proxy) is selected as the TempMaster and GP Response has to the sent.

The maximum payload length was calculated assuming unicast source routing, NWK layer protection, NO APS protection; 5B buffer was subtracted for future extensions to the GP Response command.

A.3.6.1.5.2 Bidirectional operation

The GP specification provides a way for very limited bidirectional communication with the capable GPDs. The message sequence charts for the possible interactions are depicted in the figures below: writing into GPD (Figure 77), reading out GPD attribute (Figure 78) and GPD requesting an attribute (Figure 79).

If a sink does support bidirectional communication, the following applies:

- Transmission of GPD Read Attributes command is optional;
- Reception of GPD Read Attributes Response is:
  - optional in general,
  - mandatory if transmission of GPD Read Attributes command is supported;
- Reception of GPD Request Attributes command is mandatory;
- Transmission of GPD Write Attributes command is optional.

The other direction for each of the commands above is deprecated (since that’s implemented by the GPD).

The transmission/reception of all the commands above is transparent to the proxy implementing bidirectional communication.

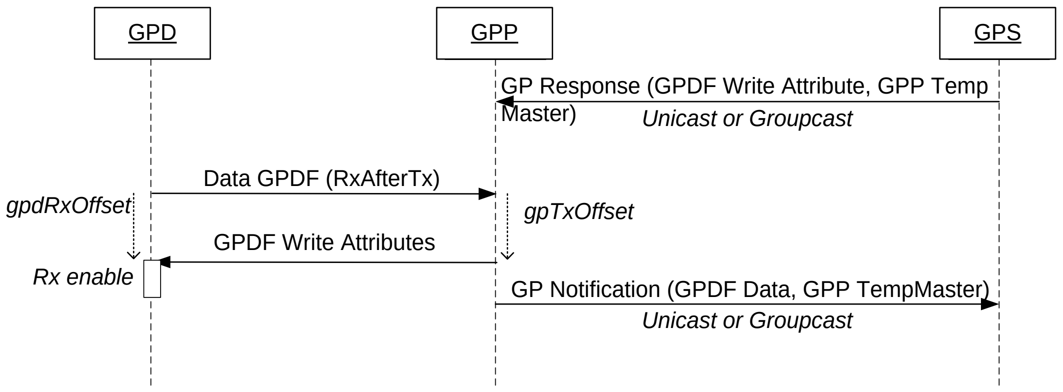


Figure 77 – MSC for GP bidirectional operation: writing into GPD

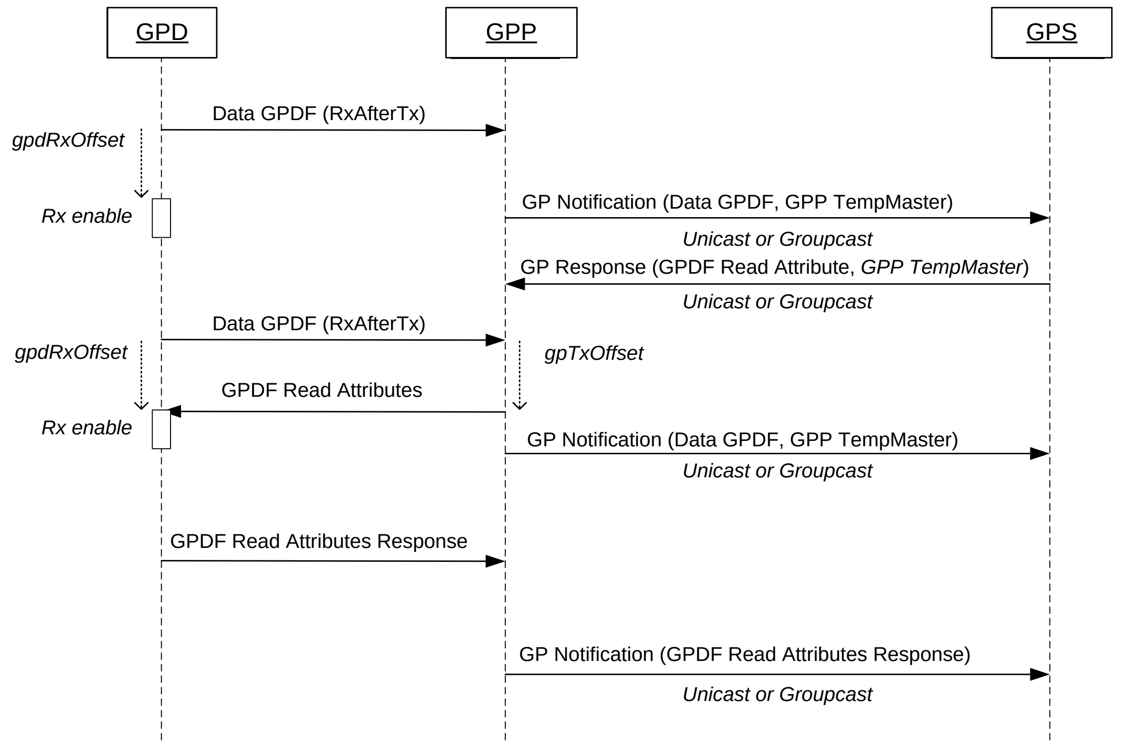

GPD
GPP
GPS
Data GPDF (RxAfterTx)
gpdRxOffset
Rx enable
GP Notification (Data GPDF, GPP TempMaster)
Unicast or Groupcast
GP Response (GPDF Read Attribute, GPP TempMaster)
Unicast or Groupcast
Data GPDF (RxAfterTx)
gpdRxOffset
gpTxOffset
GPDF Read Attributes
Rx enable
GP Notification (Data GPDF, GPP TempMaster)
Unicast or Groupcast
GPDF Read Attributes Response
GP Notification (GPDF Read Attributes Response)
Unicast or Groupcast

Figure 78 – MSC for GP bidirectional operation: reading out GPD attribute

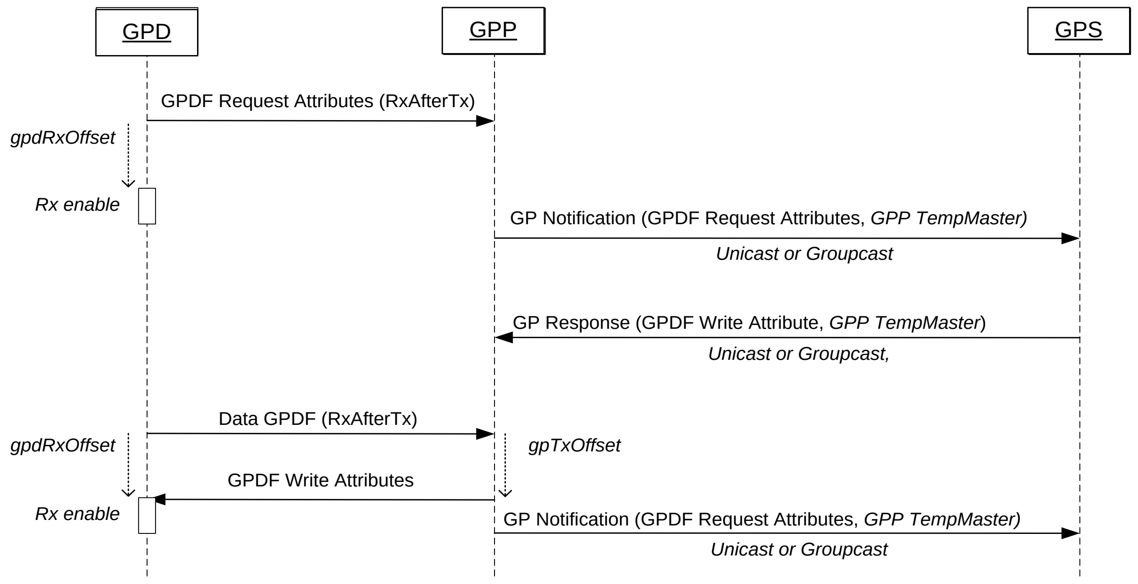

GPD
GPP
GPS
GPDF Request Attributes (RxAfterTx)
gpdRxOffset
Rx enable
GP Notification (GPDF Request Attributes, GPP TempMaster)
Unicast or Groupcast
GP Response (GPDF Write Attribute, GPP TempMaster)
Unicast or Groupcast,
Data GPDF (RxAfterTx)
gpdRxOffset
gpTxOffset
GPDF Write Attributes
Rx enable
GP Notification (GPDF Request Attributes, GPP TempMaster)
Unicast or Groupcast

Figure 79 – MSC for GP bidirectional operation: GPD requesting an attribute

A.3.6.2 Sink implementation

A.3.6.2.1 GPD application functionality matching

Implementation of GPD application functionality matching is vendor-specific.

For example, the GPD DeviceID, sent in the Commissioning GPDF, can be translated into the Zigbee DeviceID for the corresponding profile, with the list of mandatory Zigbee Clusters for that DeviceID and a Match Descriptor can be performed with the application endpoints in commissioning mode. If the *Application Information* field (see sec. A.4.2.1.1.4 - A.4.2.1.1.9) are present in the GPD Commissioning command, then the fields *GPD Command list* and *Cluster list* SHALL also be analyzed. If the *GPD Application Description command follows* sub-field of the *Application Information* field is set to 0b1, then the information in the GPD Application Description command(s) following the Commissioning GPDF SHALL also be analyzed. [175]If in the received GPD Application Description command, in any *Attribute Record* field, both the *Reported* sub-field and the *Attribute value present* sub-field of the *Attribute Options* field are set to 0b0, the sink skips that attribute and continues application functionality matching for the remainder of the frame.

Alternatively, the GPD CommandID, sent in GPD frame, can be translated into the corresponding Zigbee CommandID of a Zigbee Cluster (see sec. A.4.3), and this cluster can be bound to the application endpoints in commissioning mode.

A.3.6.2.2 GPD application functionality translation

The sink needs to translate GPD specific application functionality (GPDF device identifiers and GPD commands) relevant for sink's application endpoints into Zigbee ZCL commands. One way to solve it is to implement the Translation Table, as defined below.

Vendors of the sinks NOT using the default translations or not implementing the Translation Table functionality should think of ways how to explain the application behavior on reception of GPD commands (to the user and the testers), and how correct execution may be made observable (for the users and for certification). They MAY also provide means for controlling this functionality, other than the Translation Table.

Note: the Translation Table also finds use in other GP functionality, e.g. Sink Table-based groupcast forwarding functionality and CT-based commissioning functionality. Implementers that decide to implement any of that functionality without Translation Table SHALL find solutions to support the functionality-required operation.

If Translation Table functionality is supported, a sink contains a *GPD Command Translation Table*, each entry of which is formatted as shown in Table 48.

Implementers of this specification are free to implement the *GPD Command Translation Table* in any manner that is convenient and efficient, as long as it represents the data shown below.

Table 48 – Format of entries in the GPD Command Translation Table

Parameter name	Type	Range	Default	Description
Options	Unsigned 8-bit integer	Any valid	0x00	Options related to this table entry
GPD ID	Unsigned 32-bit Integer/IEEE address	Any valid	0xffffffff/0xffffffffffffffff	Identifier of the GPD
GPD Endpoint	Unsigned 8-bit integer	Any valid	N/A	Present if *ApplicationID* = 0b010, absent for *ApplicationID* = 0b000.

[175] Comment #8 from GP multi-sensor August PoC, Zigbee document 16-02611

Parameter name	Type	Range	Default	Description
GPD Command	8-bit bitmap	0x00 – 0xff	N/A	The GPD command to be translated
EndPoint	Unsigned 8-bit integer	0x00 – 0xff	0xff	The EndPoint for which the translation is valid.
Zigbee Profile	Unsigned 16-bit Integer	Any Valid	0xffff	The Profile of the command after translation
Zigbee Cluster	Unsigned 16-bit Integer	Any valid	N/A	The cluster of the Profile on the endpoint.
Zigbee CommandID	Unsigned 8-bits integer	Any valid	N/A	The Command ID of the Cluster into which GP Command is translated.
Zigbee Command payload	Variable	N/A	N/A	The payload for the Zigbee Command.
Additional information block	Sequence of unsigned 8-bit integer	Any valid	N/A	The information about the payload of the GPD command and other contextual information relevant for the translation

The *Options* field SHALL be formatted as shown in Figure 80.

Bits: 0..2	3	4..7
ApplicationID	Additional information block present	Reserved

Figure 80 – Format of the Options field of the GPD Command Translation Table entry

The *ApplicationID* sub-field contains the information about the application used by the GPD. *ApplicationID* = 0b000 indicates the *GPD ID* field has the length of 4B and contains the GPD SrcID; the *Endpoint* field is absent. *ApplicationID* = 0b010 indicates the *GPD ID* field has the length of 8B and contains the GPD IEEE address; the *Endpoint* field is present. All values of *ApplicationID* other than 0b000 and 0b010 are reserved in the current version of the Green Power cluster specification.

[176]The *Additional information block present* sub-field, if set to 0b1, indicates that the *Additional information block* field is present; if set to 0b0, it indicates that the *Additional information block* field is absent.

The *Zigbee Command payload* field is formatted as defined in Figure 81.

Octets	1	Variable
Data Type	unsigned 8-bit integer	Sequence of unsigned 8-bit integer
Field Name	Length	Payload

Figure 81 – Format of the Zigbee Command Payload field of the Translation Table entry

[176] LB v07: https://workspace.zigbee.org/kws/groups/zigbee_pro_foundation/comments/view_comment?comment_id=138

If the *EndPoint* field is set to 0xfd, there are no paired endpoints. If the *EndPoint* field is set to 0xff, all matching endpoints are paired. If the *EndPoint* field is set to 0xfc, the raw GPD command is passed up to the application, and no translation is performed in the GPEP.

If the *GPD Command* field is set to 0xAF, all of the following GPD sensor report commands: 0xA0 – 0xA3 are supported. Thus, 0xAF is not used as a true GPD CommandID, but as a way to make the Translation Tables more compact. [177]The *GPD Command* set to 0xAF SHALL NOT be used for translations for the GPD Compact Attribute Reporting [178]command 0xA8. If the GPD Command field is set to 0xFF, it indicates all GPD commands.

If the *Zigbee Cluster* field is set to 0xffff, the ClusterID from the triggering GPD command is to be used. If the *Zigbee Cluster* field is set to value other than 0xffff, then for GPD command carrying a *ClusterID* field (as e.g. for the GPD commands 0xA0 – 0xA3), the two ClusterID values SHALL exactly match.

If the *Length* sub-field of the *Zigbee Command payload* field is set to 0x00, the *Payload* sub-field is not present, and the Zigbee command is sent without payload. If the *Length* sub-field of the *Zigbee Command payload* field is set to 0xff, the *Payload* sub-field is not present, and the payload from the triggering GPD command is to be copied verbatim into the Zigbee command. If the *Length* sub-field of the *Zigbee Command payload* field is set to 0xfe, the *Payload* sub-field is not present, and the payload from the triggering GPD command needs to be parsed. For all other values of the *Length* sub-field, the *Payload* sub-field is present, has a length as defined in the *Length* sub-field and specifies the payload to be used.

The *Additional information block* field is formatted as defined in Figure 82.

[179]Octets	[180]1	0/Variable	…	0/Variable
Data Type	unsigned 8-bit integer	Sequence of unsigned 8-bit integer	…	Sequence of unsigned 8-bit integer
Field Name	Additional information block length	Option record 1	…	Option record N

Figure 82 – Format of the *Additional information block* field of the Translation Table entry

[181]The *Additional information block length* field carries the total length in octets of the *Additional information block*, including the length of the *Additional information block length* field, [182]decremented by one. Thus, the *Additional information block length* field set to 0x00 indicates that only octet present is the *Additional information block length* field itself.

Each *Option record* field is formatted as defined in Figure 83[1].

Octets	1	0/Variable
Data Type	unsigned 8-bit integer	Sequence of unsigned 8-bit integer
Field Name	Option selector	Option data

Figure 83 – Format of the *Option record* field of the Translation Table entry

The *Option selector* field defines the option data to follow. Each *Option selector* field is formatted as defined in Figure 86.

[177] Comment #1 from GP multi-sensor August PoC, Zigbee document 16-02611
[178] Comment #783 from GP multi-sensor v0.7 letter ballot
[179] LB v07: https://workspace.zigbee.org/kws/groups/zigbee_pro_foundation/comments/view_comment?comment_id=138
[180] Comment #711 from GP multi-sensor v0.7 letter ballot
[181] Comment #711 from GP multi-sensor v0.7 letter ballot
[182] October PoC comment #965: https://workspace.zigbee.org/kws/groups/PRO_GP/comments/view_comment?comment_id=965

Bits: 0..3	4..7
Option length	OptionID

Figure 84 – Format of the *Option selector* field of the *Option record* field of the Translation Table entry

The bits b0 – b3 of the *Option selector* field indicate the total octet length of the following *Option data* field, [183]decremented by one. Thus, *Option length* sub-field of the *Option selector* field, if set to 0x0, indicates that *Option data* field of 1 octet length follows.

The bits b4 – b7 of the *Option selector* field contain the *OptionID*. The *OptionID* sub-field defines type and format of option data to follow. The *OptionsIDs* are defined per GPD CommandID (see sec. A.3.6.2.2.1).

[184]There SHOULD be only one entry in the GPD Command Translation Table for each (GPD ID, GPD Endpoint, GPD Command, EndPoint, Zigbee Profile, Zigbee Cluster, and – if present - also the relevant part of the Additional information; what is relevant is defined per GPD Command and Option) tuple.

Note that for a single GPD ID (and *GPD Endpoint*, if *ApplicationID* = 0b010), there MAY be multiple entries, e.g. for multiple GPD commands.

Note that for a single GPD ID (and *GPD Endpoint*, if *ApplicationID* = 0b010), the same GPD Command could result in different translated Zigbee CommandIDs, for different EndPoint, Profile and Cluster values.

Note that for a single GPD ID, if *ApplicationID* = 0b010, there MAY be multiple entries, for multiple *GPD Endpoints*, even for identical GPD commands.

By default, the GPD Command Translation Table MAY contain the generic translations (mapping the GPD commands to their ZCL equivalents, see Table 54 and Table 55) for all GP-controllable application functionality. Those generic translations SHALL use *ApplicationID* = 0b000 and *SrcID* 0xffffffff; they are then applicable to those GPD commands received from any SrcID or received from a GPD with *ApplicationID* = 0b010 and any GPD IEEE address and *Endpoint*.
If no generic translations are available by default, Translation Table entries SHALL be added upon successful completion of proximity and multi-hop commissioning, and upon reception of GP Pairing Configuration leading to Sink Table entry creation (as described in A.3.5.2.5); those entries SHALL then contain the *ApplicationID* and *GPD ID* type and value of the GPD ID (and *GPD Endpoint*, matching or 0x00 or 0xff, if *ApplicationID* = 0b010) for which they are created; mapping the GPD commands to their ZCL equivalents, see Table 54 and Table 55.

If both generic and specific translation are applicable to a particular GPD command, the specific translation supersedes the generic one.

For the manufacturer-defined GPD commands (i.e. CommandIDs 0xB0 – 0xBF), if supported, the translation SHOULD store the *ManufacturerID* value in the *ProfileID* field of the Translation Table entry. The remaining fields of the Translation Table entry MAY take undefined (all ‘F’) or specific values. If the *Length* sub-field of the *Zigbee Command payload* field is set to 0xFE, a dedicated, manufacturer-defined parsing has to be implemented.

[183] October PoC comment #965: https://workspace.zigbee.org/kws/groups/PRO_GP/comments/view_comment?comment_id=965
[184] PoC comment #26 (Zigbee document 16-02601)

The GPD Command Translation Table entry can be added, overwritten or removed with the GP Translation Table Update command.

A.3.6.2.2.1 [185]OptionIDs

For the GPD 8-bit vector: press and 8-bit vector: release commands, the OptionIDs are defined in sec. A.3.6.2.2.1.1.

For the GPD supporting GPD Compact Attribute Reporting command, the *OptionIDs* are defined in sec. A.3.6.2.2.1.2.

In the current specification, there are no *OptionIDs* defined for any other GPD commands.

A.3.6.2.2.1.1 [186]OptionIDs for GPD 8-bit vector commands

For the GPD 8-bit vector: press and 8-bit vector: release commands, the *OptionID* sub-field can take any of the non-reserved values from Table 50.

Table 49 – Values of the OptionID sub-field of the Additional information field of the Translation Table entry for the GPD 8-bit vector: press/release commands187

Value	Meaning
0x0	Generic switch command execution
0x1 – 0xf	Reserved

The *Option data* of the *Generic switch command execution* option for the GPD 8-bit vector: press/release commands is formatted as defined in Figure 86. The *Generic switch command execution* option SHALL be present if the GPD Command field of the Translation Table entry is set to GPD commands 8-bit vector: press or GPD 8-bit vector: release, and its support is mandatory for the sinks implementing those commands and the Translation Table functionality.

Octets	**1**	**1**
Data Type	8-bit bitmap	8-bit bitmap
Field Name	Contact status	Contact bitmask

Figure 85 – Format of the Option data of the Generic switch command execution option of the Translation Table entry

The *Contact status* field stores the contact status values to be matched by the payload of the received GPD commands GPD 8-bit vector: press or GPD 8-bit vector: release field is to be evaluated.

The *Contact bitmask* field indicates how the *Contact status* field of the received GPD commands “GPD 8-bit vector: press” and “GPD 8-bit vector: release” is to be evaluated. An AND operation is performed taking the *Contact bitmask* and the received *Contact status* as input, and the result is compared with the *Contact status* from the Translation Table entry. If both are equal, the translation is applicable and shall be executed.

[188]If *Contact bitmask* field of a Translation Table entry is set to 0x00 then the *Contact status* field indicates all the buttons of this GPD that are paired with the current sink. This may be used for compact Translation Table representation, typically in combination with GPD processing in the application (*EndPoint* field set to 0xfc), e.g. on sinks being dynamic devices.

[185] PoC comment #10 (Zigbee document 16-02601)
[186] PoC comment #8 (Zigbee document 16-02601)
[187] PoC comment #8 (Zigbee document 16-02601)
[188] Clarification for a special case of Translation Table entry with Additional Information for GPD 8-bit vector: press command with *Contact bitmask* =

[189]For the GPD 8-bit vector press/release commands, if the *Length* sub-field of the *Zigbee Command payload* field is set to 0xfe, the *Contact status* field, if the *Contact bitmask* field is non-zero, indicates the prior state, if it is relevant to keep it.

[190]If state tracking is being performed, the sinks SHOULD NOT start the tracking with the *Current contact status* field of the GPD Commissioning command, because that contact status was transmitted for commissioning purposes and not for operational control purposes.

[191]Both the *Contact status* field and the *Contact bitmask* field SHALL be included in checking uniqueness and finding matching Translation Table entries for GPD 8-bit vector press/release commands. In addition to the generic Translation Table matching rules as defined in sec. A.3.6.2.2, if the *Length* sub-field of the *Zigbee Command payload* field is NOT set to 0xfe, the *Contact status* of the triggering GPD command is first bitwise ANDed with the *Contact bitmask* field of the Translation Table entry for the triggering GPD command of the triggering GPD, and then compared with the *Contact status* field from the Translation Table. If they are identical, a matching Translation Table entry is found.

A.3.6.2.2.1.2 OptionIDs for GPD Compact Attribute Reporting

For the GPD supporting GPD Compact Attribute Reporting command, the *OptionID* sub-field can take any of the non-reserved values from Table 50.

Table 50 – Values of the *OptionID* sub-field of the *Additional information block* field of the Translation Table entry for the GPD supporting GPD Compact Attribute Reporting command

Value	Meaning
0x0	Reportable attribute record
0x1 – 0xf	Reserved

The *Option data* part of the *Reportable attribute record* option for the GPD Compact Attribute Reporting command is formatted as defined in Figure 86. The *Reportable attribute record* option SHALL be present if the GPD Command field of the Translation Table entry is set to GPD Compact Attribute Reporting command, and its support is mandatory for the sinks implementing those commands and the Translation Table functionality.

Octets	1	1	2	2	1	1	0/2
Data Type	Unsigned 8-bit integer	Unsigned 8-bit integer	16-bit enumeration	16-bit enumeration	8-bit enumeration	8-bit bitmap	16-bit enumeration
Field Name	Report identifier	Attribute Offset within Report	ClusterID	AttributeID	Attribute Data Type	Attribute Options	Manufacturer ID

Figure 86 – Format of the *Option data* of the *Reportable attribute record* option of the Translation Table entry

The *Report identifier* field stores the values to be matched by the *Report Identifier* field in the payload of the received GPD Compact Attribute Reporting command.

The *Attribute Offset within Report* field stores the start position (in bytes) of the data point identified by the *AttributeID* of the *ClusterID* in the payload of the received GPD Compact Attribute Reporting command.

The *ClusterID* field stores the value of the ClusterID as defined in the public Zigbee ZCL [3].

0x00, as agreed during GP WG call of November 16th, 2016
[189] PoC comment #18 (Zigbee document 16-02601)
[190] PoC comment #5 (Zigbee document 16-02601)
[191] PoC comment #26 (Zigbee document 16-02601)

The *AttributeID* field stores the value of the AttributeID of the cluster indicated in the *ClusterID* field as defined in the public Zigbee ZCL [3]. The standard and manufacturer-specific attributes SHALL use appropriate AttributeIDs, as defined in Table 58.

The *Attribute Data Type* field stores the data type of the attribute that is being reported.

The *Attribute Options* field is formatted as defined in Figure 86.

Bits: 0	1	2..7
[192]Client / server	ManufacturerID present	Reserved

Figure 87 – Format of the *Attribute options* field *of the Reportable attribute record* option of the Translation Table entry

[193]The *Client / server* sub-field is a Boolean flag. If set to 0b1, it indicates the GPD implements the server side of the cluster identified by the *ClusterID* field. If set to 0b0, it indicates the GPD implements the client side of the cluster identified by the *ClusterID* field.

The *ManufacturerID present* sub-field is a Boolean flag. If set to 0b1, it indicates that the *ManufacturerID* field is present. If the *ClusterID* is from a manufacturer-specific range, as defined in the Zigbee ZCL [3], or if the *AttributeID* is from the Green Power manufacturer-specific attribute range, as defined in Table 58, the attribute is manufacturer-specific; otherwise the attribute as indicated by the *AttributeID* field is a standard attribute of the cluster identified by *ClusterID* as defined in the ZCL [3].The *ManufacturerID* field, if present, stores the manufacturer code as defined in [7].

A.3.6.2.2.2 Default recommended execution rules

A.3.6.2.2.2.1 Default recommended execution rules for GPD 8-bit vector commands

If a sink supports the reception of GPD 8-bit vector commands and is a simple device (see the definition in [15]), it SHALL support default execution rules for the GPD 8-bit vector commands. Those execution rules can be encoded as Translation Table entries, if the Translation Table feature is supported; then, they can also be reconfigured over the air, using the Translation Table commands.

The current specification provides default recommended execution rules which represent the most prevalent usage of generic switches to-date in the market. Different execution rules MAY be implemented, depending on the sink application functionality.

It is assumed every button or rocker side corresponds to a single contact, which is represented on a single bit.

Table 51 specifies default recommended translation for a sink being a dimmable light.

Table 51 – Default recommended translations for sink being a dimmable light

Switch type	Number of contacts (bits) paired with the sink	Default recommended translation at the sink
Generic, Button	1	The bit is interpreted as a TOGGLE command; the corresponding release bit is ignored
	2	The first bit (or higher bit, in case of simultaneous activation during commissioning) is interpreted as ON command

[192] Comment #2 from GP multi-sensor August PoC, Zigbee document 16-02611
[193] Comment #2 from GP multi-sensor August PoC, Zigbee document 16-02611

		The second (lower) bit is interpreted as OFF command The corresponding release bits are ignored
	3	The second bit (or lowest bit, in case of simultaneous activation during commissioning) is interpreted as MOVE DOWN command and the corresponding release bit as STOP The first (middle) bit is interpreted as MOVE UP command and the corresponding release as STOP The third (highest) bit as a TOGGLE command; the corresponding release bit is ignored
	4	The second bit (or lowest bit, in case of simultaneous activation during commissioning) is interpreted as OFF command; the corresponding release bit is ignored The first (lower middle) bit is interpreted as ON command; the corresponding release bit is ignored The fourth (higher middle) bit is interpreted as MOVE DOWN command and the corresponding release bit as STOP The third (highest) bit is interpreted as MOVE UP command and the corresponding release as STOP
	5 and more	No recommended default translation
Rocker	1 (or both from the same rocker)	As for 2-button switch above
	2, being at least one (or both) from each rocker)	As for 4-button switch above
	3 or more rockers	No recommended default translation

Table 52 specifies default recommended translation for a sink being a blinds controller.

Table 52 – Default recommended translations for sink being a blinds controller

Switch type	Number of contacts (bits) paired with the sink	Default recommended translation at the sink
Generic, Button	1	No recommended default translation
	2	The first bit (or higher bit, in case of simultaneous activation during commissioning) is interpreted as MOVE UP command and the corresponding release bit as STOP The second (lower) bit is interpreted as MOVE DOWN command and the corresponding release as STOP
	3	The first bit (or middle bit, in case of simultaneous activation during commissioning) is interpreted as MOVE UP command The second (lowest) bit is interpreted as MOVE DOWN command The third (highest) bit as a STOP command; The corresponding release bits are ignored
	4	No recommended default translation
	5 and more	No recommended default translation
Rocker	1 (or both from the same rocker)	As for 2-button switch above
	2, being at least one (or both) from each rocker)	No recommended default translation
	3 or more rockers	No recommended default translation

[194]During commissioning, a sink [195]SHOULD only store the bits of the *Current contact status* field of the Commissioning GPDF that correspond to the *Number of contacts* of the *Generic switch configuration* field; any higher bits set in the received *Current contact status* MAY be zeroed before storing; any Commissioning GPDF carrying *Current contact status* field in which only bits higher than the *Number of contacts* are set to 0b1 SHOULD be silently dropped.

[194] LB v07: https://workspace.zigbee.org/kws/groups/zigbee_pro_foundation/comments/view_comment?comment_id=309
[195] PoC comment #6 (Zigbee document 16-02601)

A.3.6.2.3 TempMaster election

Within *Dmax* ms (see A.3.6.3.1) after the reception of the first instance of this command, the sink creates a list of candidate responders, consisting of the proxies which did forward GP (Commissioning) Notification command with the *BidirectionalCommunicationCapability* sub-field of the *Options* field set to 0b1, if any, *gpTxQueueFull* sub-field of the *Options* field set to 0b0, if any, as well as itself, if it did receive the GPD command directly.

If the sink is in operational mode and there were NO candidates supporting bidirectional communication (i.e. for all candidates the *BidirectionalCommunicationCapability* sub-field of the *Options* field was set to 0b0), the sink SHALL abandon the TempMaster election and the attempted transmission.

If (i) the sink is in commissioning mode, and there were NO candidates supporting bidirectional communication (i.e. for all candidates the *BidirectionalCommunicationCapability* sub-field of the *Options* field was set to 0b0) or (ii) the sink is in operation and there are candidates capable of bidirectional communication, the sink SHALL select from the available candidates with *BidirectionalCommunicationCapability* sub-field of the *Options* field set to 0b1, as follows.

The sink selects the node with the best *GPP-GPD link* value for this GPD (and *Endpoint*, if *ApplicationID* = 0b010 and the sink selects *Transmit on endpoint match* = 0b1), whereby better *GPP-GPD link* is defined as one having higher value of the *Link quality* sub-field, and if *Link quality* is equal, as one having higher value of the *RSSI* sub-field; or if multiple have the same *GPP-GPD link* value, the one with the best *GPP-GPD link* value and lowest short address.

If another device is chosen as the TempMaster, the sink sends the GP Response frame carrying the APPL data payload (*GPD CommandID* and *GPD Command Payload*) to be transmitted to GPD. The GP Response SHOULD be sent in broadcast, and it SHALL then carry the short address of the selected TempMaster in the *TempMaster short address* of the payload; it MAY be sent in unicast to the TempMaster instead.

If the sink itself is chosen as the TempMaster, it SHOULD broadcast the GP Response, and it SHALL then carry the short address of the sink in the *TempMaster short address* of the payload.

A.3.6.2.4 [196]*MultiSensorCommissioningTimeout*

A sink supporting any functionality controllable via GPD Compact Attribute Reporting command and the CT-based commissioning feature SHALL support the *MultiSensorCommissioningTimeout.*

The *MultiSensorCommissioningTimeout* is used to time-limit the CT-based commissioning of a GPD supporting GPD Compact Attribute Reporting, in order to check the completeness of the buffered commissioning information.

The *MultiSensorCommissioningTimeout* SHALL have a value of 20s.

A.3.6.2.5 *MultiSensorCommissioningBufferSize*

A sink supporting any functionality controllable via GPD Compact Attribute Reporting command and the CT-based commissioning functionality and Pre-commissioned groupcast functionality SHALL support the *MultiSensorCommissioningBufferSize.*

The *MultiSensorCommissioningBufferSize* defines the minimum number of complete GP Pairing Configuration command with *Action* sub-field of the *Actions* field set to 0b101 (application description), i.e. carrying the Report Descriptors, that the sink SHALL be capable of storing to forward to the other group members upon successful pairing.

The *MultiSensorCommissioningBufferSize* SHALL have a value of 1.

[196] GP multi-sensor LB v0.9 comment #973: https://workspace.zigbee.org/kws/groups/PRO_GP/comments/view_comment?comment_id=973

A.3.6.3 Proxy implementation

A.3.6.3.1 gppTunnelingDelay,

The gppTunnelingDelay is the time between the reception of a GPDF by a proxy-capable device and forwarding of a GP Notification or GP Commissioning Notification or a GP Tunneling Stop carrying the GPD command from the GPDF.

The gppTunnelingDelay is calculated, taking into account the following criteria:

- whether the received GPDF had the *RxAfterTx* sub-field set;
- *Link quality* to the GPD, as reported in GP (Commissioning) Notification (see sec. A.3.3.4.1);
- Only if full unicast communication mode in operation is used:
  - knowledge of the route to the GP sink;
  - Fact of being first to forward for the previous GPDF from this GPD.

The gppTunnelingDelay can be calculated according to the following formula

$$\text{gppTunnelingDelay [ms]} = \begin{cases} \text{Dmin;} & \text{if FirstToForward = TRUE \& NoRoute=FALSE} \\ \text{Dmin + QualityBasedDelay;} & \text{if FirstToForward = FALSE \& NoRoute=FALSE} \\ \text{Dmin + Dmax;} & \text{if NoRoute=TRUE} \end{cases}$$

where:

- Dmin =
  - if the triggering GPDF had *RxAfterTx* = 0b0: Dmin_u = 5 ms;
  - if the triggering GPDF had *RxAfterTx* = 0b1: Dmin_b = 32 ms;
- QualityBasedDelay is calculated as follows:
  - For *Link quality* = 0b11: 0 ms;
  - For *Link quality* = 0b10: 32ms;
  - For *Link quality* = 0b01: 64ms;
  - For *Link quality* = 0b00: 96ms;
- Dmax=100ms
- NoRoute is a Boolean flag: as stored in the Proxy Table entry for this GPD; this is only taken into account if full unicast communication mode in operation is used.
- FirstToForward is a Boolean flag, as stored in the Proxy Table entry for this GPD; this is only taken into account if full unicast communication mode in operation is used.

Note that for any communication mode, the Zigbee stack adds additional randomized delays.

The gppTunnelingDelay is intended to indicate the time as measured on the medium. If the delay introduced by the stack can be estimated, it can be taken into account for the gppTunnelingDelay calculation at the Green Power EndPoint.

A.3.6.3.2 gppCommissioningWindow

The default value is 180 seconds.
The default value for the proxy, *gppCommissioningWindow*, can be overwritten by the sink for the duration of one particular commissioning procedure, by including the *CommissioningWindow* field in the GP Proxy Commissioning Mode message.

A.3.6.3.3 Proxy aliasing

A sink is capable of filtering the GP (Commissioning) Notification commands at the Green Power EndPoint level. However, multiple proxies tunneling the same GPDF in groupcast mode would result in a lot of (unnecessary) network traffic and clog the NWK BTTs of all routers.

To allow also the lower layers (NWK) of the other proxy and router devices, as well as of the sinks, to filter the messages sent by the proxies on behalf of the same GPD, the proxies originating the message use – in certain cases defined by the current specification - proxy aliasing, i.e. Alias NWK level source short address and Alias NWK level sequence number.

Note, that there is a certain, network-size dependent probability of two different GPD IDs resulting in the same derived alias source address. As long as the alias sequence numbers are different, the Green Power EndPoint will be able to filter out, based on the full GPD ID (and *Endpoint*, if *ApplicationID* = 0b010) in the GP Notification payload. There is also a certain probability of the two derived alias source addresses being simultaneously used with the same sequence number, but it is considered negligible.

In addition, to prevent that subsequent GP (Commissioning) Notification commands, especially if forwarded by different proxies, coincidentally use the same APS counter value thus leading to GP command dropping by the APS duplicate rejection table of the receiving sink, if proxy aliasing is used, the APS counter of the transmitted Green Power cluster command takes the value of the alias sequence number.

A.3.6.3.3.1 Derivation of alias source address

If no *Assigned Alias* is stored in the Proxy Table entry for a particular GPD, the Alias NWK level source short address, Alias_src_addr, is derived from the GPD ID in the following way, the same for *ApplicationID* 0b000 and 0b010; If *ApplicationID* = 0b010, the *Endpoint* field SHALL NOT be used for alias derivation.

The 2 LSB of the GPD ID are examined. If they do not correspond to any of the reserved Zigbee short addresses (0x0000 for the Zigbee Coordinator, and the addresses exceeding 0xfff7, reserved for broadcasts), this value is used as Alias_src_addr. Otherwise, if the resulting Alias_src_addr does correspond to one of the reserved Zigbee short addresses, the 2 LSBs of the GPD ID SHALL be XORed with the 3rd and 4th LSB of the GPD ID, i.e. 1st LSB XORed with 3rd LSB and 2nd LSB XORed with 4th LSB. If the resulting value does not correspond to any of the reserved Zigbee short addresses, this value is used as Alias_src_addr. Otherwise, if the XORed value corresponds to a reserved Zigbee short address, then in case the 2 LSB of the GPD ID were 0x0000, a value of 0x0007 SHALL be used, or else the value of 0x0008 SHALL be subtracted from the 2 LSB.

A.3.6.3.3.2 Derivation of alias sequence number

The proxies use the Alias NWK level sequence number and Alias APS counter which – both for assigned and derived alias - have the identical value derived from MAC header sequence number of the trigger GPDF. Specifically:

- The derived groupcast GP Notification command uses the exact value from the GPDF MAC header *Sequence number* field;
- The GP Pairing Search command uses the value: GPDF_MAC_header_*Sequence_number* – 10 (mod 256);
    - Note: if the transmission of the GP Pairing Search command was triggered by reception of another GP command (e.g. GP Notification or GP Tunneling Stop), the correct sequence number needs to be derived from the information available in this frame.
      E.g. if the trigger was GP Tunneling Stop, then the alias sequence number to be used for GP Pairing Search is to be calculated as follows:

GP_Tunneling_Stop_NWK_header_*Sequence_number* +1.

  - if the transmission of the GP Pairing Search command was not triggered by reception of GPD command, and thus the current GPD MAC *Sequence number* value for this GPD is not available, a random value SHOULD be used.

- The GP Tunneling Stop command uses the value: GPDF_MAC_header_*Sequence_number* – 11 (mod 256);
- The GP Commissioning Notification command uses the value: GPDF_MAC_header_*Sequence_number* – 12 (mod 256);
- The commissioned groupcast GP Notification command uses the value: GPDF_MAC_header_*Sequence_number* – 9 (mod 256);
- The broadcast GP Notification command uses the value: GPDF_MAC_header_*Sequence_number* – 14 (mod 256);
- The Device_annce command uses the value of 0x00.

A.3.6.3.4 Alias use vs. regular Zigbee

A.3.6.3.4.1 Sending Device_annce on behalf of GPD

There is a certain, network-size dependent probability of address conflict between the GPD ID-derived alias and genuine randomly assigned Zigbee NWK address. SHOULD this be detected, it is expected to be resolved by the Zigbee device changing its unique address, as specified by the Zigbee protocol.

To assure that usage of the alias does not cause any disturbance to Zigbee network operation, the sink SHALL send the Zigbee Device_annce command [1], after adding an active entry for a new GPD into its Sink Table as a result of proximity or multi-hop commissioning (see sec. A.3.9.1).

[197]A GP CT SHOULD send the Zigbee Device_annce command [1], when adding an active Proxy Table entry using GP Pairing command with *AddSink* sub-field of the *Options* field set to 0b1 or a Sink Table entry using GP Pairing Configuration command with *Send GP Pairing* sub-field of the *Actions* field set to 0b0, i.e. when the Device_annce will not be sent by the sink; when multiple entries for the same GPD are added at the same time, it is sufficient to send Device_annce once.

[198]In addition, a sink and a GP CT MAY also send Device_annce at other times, e.g. to prevent/resolve conflicts with devices not present at the time of the original announcement.The proxy SHALL NOT send Device_annce in commissioning mode.

When the proxy is in operational mode and observes a GPDF for which the security check fails and for which GPD ID it does not have a Proxy Table entry, the proxy SHALL NOT send Device_annce and SHALL NOT use the alias, until the GPD's membership in the network is confirmed.

A.3.6.3.4.2 Format of Device_annce sent on behalf of GPD

The Zigbee Device_annce command SHALL always be sent using the Alias source address as NWK source address, a fixed NWK sequence number of 0x00, and a fixed APS counter of 0x00.

The payload of the Zigbee Device_annce command SHALL carry the following information the same for *ApplicationID* 0b000 and 0b010: the NWKAddr field SHALL carry the alias for the GPD, either the calculated Alias NWK source address (see sec. A.3.6.3.3) or the AssignedAlias; the IEEEAddr field SHALL carry the 0xffffffffffffffff value indicating invalid IEEE address [3], and the Capability field with the values as indicated in Figure 88.

[197] CCB #2408; resolution modified in 15-02014-013 as a result of Kavi comment #1378 from letter ballot for GP Bsic errata set: https://workspace.zigbee.org/higherlogic/ws/groups/PRO_GP/comments/view_comment?comment_id=1378
[198] CCB #2408; resolution modified in 15-02014-013 as a result of Kavi comment #1378 from letter ballot for GP Bsic errata set: https://workspace.zigbee.org/higherlogic/ws/groups/PRO_GP/comments/view_comment?comment_id=1378

Bits: 0	1	2	3	4-5	6	7
Alternate PAN coordinator	Device type	Power source	Receiver on when idle	Reserved	Security capability	Allocate address
0	0	0	0	00	Inherited from the proxy	0

Figure 88 – Values for the Capability field of the Zigbee Device_annce command, sent by the proxies on behalf of the Alias NWK address

A.3.7 GP security

A.3.7.1 Implementation

A.3.7.1.1 Security parameters

The dGP stub of a proxy SHALL support all security levels defined in the GP specification.

The dGP stub of a sink SHALL support all security levels above and including the application- and product-specific minimum security level, as indicated in the *gpsSecurityLevel* attribute.

A.3.7.1.2 gpSecurityKeyType

The *gpdSecurityKeyType* can take the values as defined in Table 53.

Table 53 – Values of gpSecurityKeyType

Value	Description	Comment	Security properties
0b000	No key		No protection for GPDF communication. The attacker can eavesdrop and spoof all GPDF communication.
0b001	Zigbee NWK key	The Zigbee Network key (as stored in the NIB *Key* parameter) is used for securing the communication with the GPD. Thus, the key is readily available to any proxy/sink being part of the Zigbee network. It needs to be delivered to any security-capable GPD. Note: in the event of NWK key update, updating the key on the GPDs is required as well.	Overhearing in the clear key transmission/compromising one GPD compromises the Zigbee NWK key, which allows the attacker to eavesdrop and spoof all Zigbee and GP communication and all the devices of the entire Zigbee network.
0b010	GPD group key	Group key is shared between GPDs and GP infrastructure devices. The key is needs to be configured into all GP infrastructure devices and all security-capable GPDs.	Overhearing in the clear key transmission /compromising one GPD allows the attacker to eavesdrop and spoof all GPDF communication. However, it does not allow the attacker to add new GPDs, thanks to the dedicated commissioning of GPD into the network.
0b011	NWK-key derived GPD group key	Group key is shared between GPDs and GP infrastructure devices, which is derived from the Zigbee Network key as specified in A.3.7.1.2.1. Thus, the key is readily available to any proxy/sink being part of the Zigbee network. Only the derived key - and not the NWK key - is delivered to any GPD. Note: in the event of NWK key update, updating the key on the GPDs is required as well.	Overhearing in the clear key transmission/compromising one GPD allows the attacker to eavesdrop and spoof all GPDF communication. However, because of the properties of the derivation function (see A.3.7.1.2.1), it does not reveal the Zigbee NWK key. It also does not allow the attacker to add new GPDs, thanks to the dedicated commissioning of GPD into the network.
0b100	(individual) out-of-the-box GPD key	GPD is pre-configured with a security key. The key is needs to be configured into all (relevant) GP infrastructure devices.	Overhearing in the clear key transmission /compromising one GPD does allow the attacker to eavesdrop/spoof any communication of this particular device. It does not give the attacker any additional benefit.

Value	Description	Comment	Security properties
0b101-0b110	Reserved		
0b111	Derived individual GPD key	An individual key is derived from the GPD independent group key (0x010) used by a particular network, as specified in sec. A.3.7.1.2.2. When the Derived individual GPD key type is used, the *gpSharedSecurityKeyType* attribute SHALL store the value 0b111, and the *gpSharedSecurityKey* attribute SHALL store the value of the GPD group key (0b010). Only the derived key (and not the shared key) is delivered to any GPD.	Overhearing in the clear key transmission/compromising one GPD allow the attacker to eavesdrop/spoof any communication of this particular device. However, because of the properties of the derivation function (see sec. A.3.7.1.2.2), it does not reveal the shared key. It does not allow the attacker to add new GPDs, thanks to the dedicated commissioning of GPD into the network.

A.3.7.1.2.1 GPD group key (0b011) derivation

The HMAC keyed hash function, as defined in [17], is used to derive the GPD group key (0b011).

$$K_{GP}= HMAC(K, ‘ZGP’)_{16}$$

whereby

- the block size *B*, the length of the key *K* and the output size *t* (of the GPD group key K_{GP}) are all 128 bit/16 octets;
- the Matyas-Meyer-Oseas hash function, as defined in [1] section B.6, is used as the hash function *H*;
- the character string ‘Z’ ‘G’ ‘P’ is used as the *text* input, with each ASCII character represented on 8bit;
- the Zigbee NWK key is used as the key *K*.

Implementation of key derivation is only mandatory for the sink; the proxies receive the correct key in the GP Pairing command.

A.3.7.1.2.2 Individual GPD key derivation

The HMAC keyed hash function, as defined in [17], is used to derive the individual GPD key.

$$K_{GPD\ ID}= HMAC(K, ID)_{16}$$

whereby

- the block size *B*, the length of the key *K* and the output size *t* (of the individual key $K_{GPD\ ID}$) are all 128 bit/16 octets;
- the Matyas-Meyer-Oseas hash function, as defined in [1] section B.6, is used as the hash function *H*;
- the ID is:
  - for GPD using *ApplicationID* = 0b010, i.e. identified by IEEE address: 8B GPD IEEE address is used as the *text* input, in little endian order (e.g. 0x11 0xff 0xee 0xdd 0xcc 0xbb 0xaa 0x00 for IEEE address 00:aa:bb:cc:dd:ee:ff:11); the *Endpoint* field SHALL NOT be used;
  - for GPD using *ApplicationID* = 0b000, i.e. identified by SrcID: 4B GPD SrcID is used as the *text* input, in little endian order (e.g. 0x21 0x43 0x65 0x87 for SrcID=0x87654321);
- the GPD group key (0x010) as stored in the *gpSharedSecurityKey* attribute (see sec. A.3.3.3.2) is used as the key *K*.

Implementation of key derivation is only mandatory for the sink; the proxies receive the correct key in the GP Pairing command.

A.3.7.1.2.3 Over-the-air protection of GPD key with TC-LK

When the device is capable of exchanging the GPDkey field protected, it SHALL calculate the values of the GPDkey and GPDkeyMIC fields by invoking CCM* as for security Level 0b11, with the following inputs:

- Payload = GPDkey in the clear;
- Header:
  - For GPD using *ApplicationID* = 0b000: the GPD SrcID;
  - For GPD using *ApplicationID* = 0b010: 4LSB of the GPD IEEE address; the *Endpoint* field SHALL NOT be used;

    Note: the Header octets are only used for CCM* security processing; they are not included in the data transmitted over the air.
- Nonce with:
  - *Source address* parameter taking the value:
    - For GPD using *ApplicationID* = 0b000:
      - {SrcID || SrcID}, for GPDF sent by GPD;
      - {0x00000000 || SrcID}, for GPDF sent to GPD;
    - For GPD using *ApplicationID* = 0b010:
      - IEEE address of the GPD, for both GPDF send by and to GPD; the *Endpoint* field SHALL NOT be used.
  - *Frame counter* parameter SHALL take the value:
    - For GPD using *ApplicationID* = 0b000 and GPDF sent by GPD: 4B SrcID;
    - For GPD using *ApplicationID* = 0b010 and GPDF sent by GPD: 4LSB of GPD IEEE address;
    - For GPD using *ApplicationID* 0b000 or 0b010 and GPDF sent to GPD: Current_Security_frame_counter+1 (where Current_Security_frame_counter is the value from the GPDF that triggers Commissioning Reply *creation*, not *sending*); the *Endpoint* field SHALL NOT be used.
  - *Security control* field set as follows (as described in sec. A.1.5.3.2):
    - Security level (according to [1])= 0b101
    - Key identifier (NOT according to [1]) = 0b00
    - Note that this security level and Key identifier are never transmitted and are NOT used for determining the transformation applied to the packet, since those are governed by the *Security* sub-field of the NWK Frame Control field of the GPDF. The values here are defined for interoperability only.
    - Extended nonce =0b0;
    - Reserved =
      - For *ApplicationID* = 0b000 and/or for incoming secured GPDF (i.e. GPDF sent by GPD): *Reserved* = 0b00;
      - For outgoing secured GPDF (i.e. GPDF sent to GPD) with an *ApplicationID* = 0b010: *Reserved* = 0b11.

A.3.7.1.2.4 Key use recommended practices

The following key types SHALL NOT be used in any network at the same time:

- NWK key and NWK-key derived GPD group key;
- Shared key and shared-key derived individual keys.

Any of the following key types: NWK key, GP group key, derived individual keys can be used in combination with the GPD OOB individual keys.

A.3.7.2 Security assumptions

Four security levels for GPDF frame protection are offered by the specification, as summarized in Table 11. The manufacturers of the Green Power Sink devices are responsible for selecting the appropriate minimum security level required by their device type and application context it is expected to work with; by setting the *gpsSecurityLevel* attribute. The process of creating the pairings assures that sinks can only be controlled by GPDs with matching (security) capabilities.

Two-step security processing of the incoming GPDF is performed: proxies authenticate and check the freshness of the frame, before forwarding; and the sink(s) check the required security level and frame freshness before execution.

All proxy and sink nodes, as members of the Zigbee network, are assumed to be trusted.

The *SecurityLevel* 0b00 provides no protection for the GPDF itself. Still, the receiving devices are expected to check if they have a Proxy/Sink Table entry for the GPD ID. This level only protects the system on runtime against genuine non-malicious devices which were not paired to this network, e.g. neighbor's GPDs. While this level of protection is extremely low, it is considered sufficient for some applications, given the design constraints of the energy-harvesting GPDs. The decision if to support this mode is left to the sink vendors.

The *SecurityLevel* 0b10 and 0b11 provide security protection for the GPDF identical to that of Zigbee security level 0x01 and 0x05, respectively (see Table 4.38 of [1]).

In case of bidirectional communication, to simplify the counter management on the GPD, the responding GP infrastructure device (proxy, sink or combo) SHALL also use the same frame counter value as the last one used by the GPD. The uniqueness of the nonce is assured by using different value for the *Source address* field of the Nonce for sending to and from the GPD.

A.3.7.3 Security operation

A.3.7.3.1 [199]Direct communication

A.3.7.3.1.1 [200]Incoming frames

On reception of GP-SEC.request, the device SHALL check if the frame is not a duplicate, as described in A.3.6.1.2. If the frame is a duplicate, the device generates GP-SEC.response, with the Status DROP_FRAME.

If the frame is not a duplicate, the device acts differently, dependent on whether it is a sink (GPT+ or combo), see sec. A.3.7.3.1.2, or a proxy, see sec. A.3.7.3.1.3.

If the device is a combo, i.e. has both sink and proxy functionality, the Sink Table SHALL be consulted first, see sec. A.3.7.3.1.2. Whenever the security-related parameters in a Sink Table entry for a particular GPD are updated, the changes SHALL be automatically propagated to the Proxy Table.

A.3.7.3.1.2 [201]Sink

The sink (i.e. GPT+ and combo) checks if it has a Sink Table entry for this GPD.

If there no Sink Table entry for this GPD and the sink is in operational mode, and the sink is a GPT+, it SHALL generate GP-SEC.response with the Status DROP_FRAME.

If there no Sink Table entry for this GPD and the sink is in operational mode, and the sink is a combo, it SHALL act a described in A.3.7.3.1.3.

[199] CCB #2120; Resolution added in 15-02014-002
[200] CCB #2120; Resolution added in 15-02014-002
[201] CCB #2120; Resolution added in 15-02014-002

If there no Sink Table entry for this GPD and the sink is in commissioning mode and the KeyType as indicated in GP-SEC.request was 0b0, the sink fetches the shared key. If there is none, sink generates GP-SEC.response, with the Status DROP_FRAME. If there is, the sink generates GP-SEC.response, with the Status MATCH, and includes the key, the key type and the frame counter as processed here. If there is no Sink Table entry for this GPD and the sink is in commissioning mode and the KeyType as indicated in GP-SEC.request was 0b1, the sink generates GP-SEC.response, with the Status DROP_FRAME.

If there is a Sink Table entry for this GPD (note: if *ApplicationID* = 0b010, the Sink Table entry may contain a different value of the *Endpoint* parameter than that supplied by GP-SEC.request), the Sink checks the freshness of the frame and whether the SecurityLevel and SecurityKeyType from the GP-SEC.request match those from the Sink Table entry; for *SecurityKeyType* mapping Table 12 is to be used. If any of those checks fails, the sink generates GP-SEC.response, with the Status DROP_FRAME. If the checks are successful, the sink checks if the *Endpoint* parameter of the GP-SEC.request matches that in the Sink Table entry. If yes, the sink generates GP-SEC.response, with the Status MATCH, and includes the key, the key type and the frame counter as processed here. If not, the sink generates GP-SEC.response with the Status TX_THEN_DROP and includes the key, the key type and the frame counter as processed here; if the sink does not support bidirectional communication it MAY return the Status DROP instead.

A.3.7.3.1.3 [202]Proxy

The proxy checks if it has a Proxy Table entry for this GPD.

If the proxy has an active entry (note: if *ApplicationID* = 0b010, the Proxy Table entry may contain a different value of the *Endpoint* parameter than that supplied by GP-SEC.request), the proxy checks the freshness of the frame and whether the *SecurityLevel* and *SecurityKeyType* from the GP-SEC.request match those from the Proxy Table entry; for *SecurityKeyType* mapping Table 12 is to be used. If any of those checks fails, and the proxy is in the operational mode, the proxy generates GP-SEC.response, with the Status DROP_FRAME. If any of those checks fails, and the proxy is in the commissioning mode, the proxy generates GP-SEC.response, with the Status PASS_UNPROCESSED. If the checks are successful, the proxy checks if the *Endpoint* parameter of the GP-SEC.request matches that in the Proxy Table entry. If yes, the proxy generates GP-SEC.response, with the Status MATCH, and includes the key, the key type and the frame counter as processed here. If not, the proxy generates GP-SEC.response with the Status TX_THEN_DROP and includes the key, the key type and the frame counter as processed here; if the proxy does not support bidirectional communication it MAY return the Status DROP instead.

If the proxy has an inactive entry and is in operational mode, it updates the SearchCounter and generates GP-SEC.response, with the Status DROP_FRAME.

If (i) the proxy has an inactive entry and is in commissioning mode or if there is no Proxy Table entry for this GPD and (ii) the KeyType as indicated in GP-SEC.request was 0b0, the proxy fetches the shared key. If the key type was 0b1 or the key type was 0b0 and there is no shared key, proxy generates GP-SEC.response, with the Status PASS_UNPROCESSED.

A.3.7.3.1.4 [203]Incoming frames: key recovery

- If the KeyType field of the GP-SEC.request had the value of 0b1:
  - And the KeyType sub-field of the Sink/Proxy entry has the value 0b100:
    - use the GPD key stored in the Sink/Proxy Table entry for this GPD,

[202] CCB #2120; Resolution added in 15-02014-002
[203] CCB #2120; Resolution added in 15-02014-002

    - if none is stored: return DROP_FRAME.
  - And the KeyType sub-field of the Sink/Proxy entry has the value 0b111:
    - use the GPD key stored in the Sink/Proxy Table entry for this GPD
    - or if none stored in the Sink/Proxy Table entry: the individual key, derived from the *gpSharedSecurityKey*.
    - else: return DROP_FRAME.
- If the KeyType field of the GP-SEC.request had the value of 0b0:
  - And the KeyType sub-field of the Sink/Proxy entry has the value 0b001:
    - use the GPD key stored in the *gpSharedSecurityKey*, if the *gpSharedSecurityKeyType* = 0b001,
    - or the key from the Key field of the *nwkSecurityMaterialSet* NIB parameter.
    - else: return DROP_FRAME.
  - And the KeyType sub-field of the Sink/Proxy entry has the value 0b010:
    - use the GPD key stored in the *gpSharedSecurityKey*, if the *gpSharedSecurityKeyType* = 0b010,
    - else: return DROP_FRAME.
  - And the KeyType sub-field of the Sink/Proxy entry has the value 0b011:
    - use the GPD key stored in the *gpSharedSecurityKey*, if the *gpSharedSecurityKeyType* = 0b011,
    - or the key derived from the *gpSharedSecurityKey*,
    - else: return DROP_FRAME.

A.3.7.3.2 [204]Tunneled communication: sink

On reception of GP Commissioning Notification command with *SecurityProcessingFailed* sub-field of the *Options* field set to 0b1, thus carrying encrypted *GPD CommandID* and *GPD Command payload*, and the corresponding *MIC* field, the sink takes the following values to reconstruct the *Frame Control* field and *Extended Frame Control* field, required for decryption:

- Sub-fields of the *Frame Control* field:
  - *Frame type* = 0b00 (since according to the current specification, a Maintenance GPDF cannot use security);
  - *Zigbee Protocol Version* = 0x3 (fixed value);
  - *Auto-Commissioning* = 0b0 (according to the current specification);
  - *NWK Frame Control Extension* = 0b1 (implicit, since security was used);
- Sub-fields of the *Extended Frame Control* field:
  - *ApplicationID* sub-field is copied from the *ApplicationID* sub-field of the *Options* field of the GP Commissioning Notification;
  - *SecurityLevel* sub-field is copied from the *SecurityLevel* sub-field of the *Options* field of the GP Commissioning Notification;
  - *SecurityKey* sub-field is derived from the *SecurityKeyType* sub-field of the *Options* field of the GP Commissioning Notification (see Table 12);
  - *RxAfterTx* sub-field is copied from the *RxAfterTx* sub-field of the *Options* field of the GP Commissioning Notification;
  - *Direction* = 0b0 (implicit; GPD frames sent to the GPD are not forwarded).

Figure 89 below illustrates this derivation.

[204] CCB #2120; Resolution added in 15-02014-002

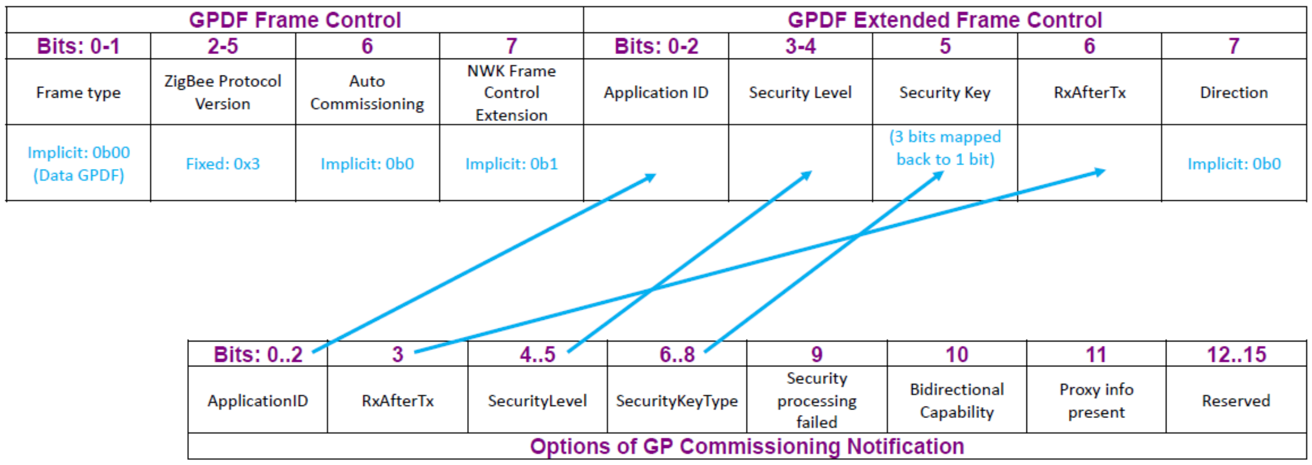
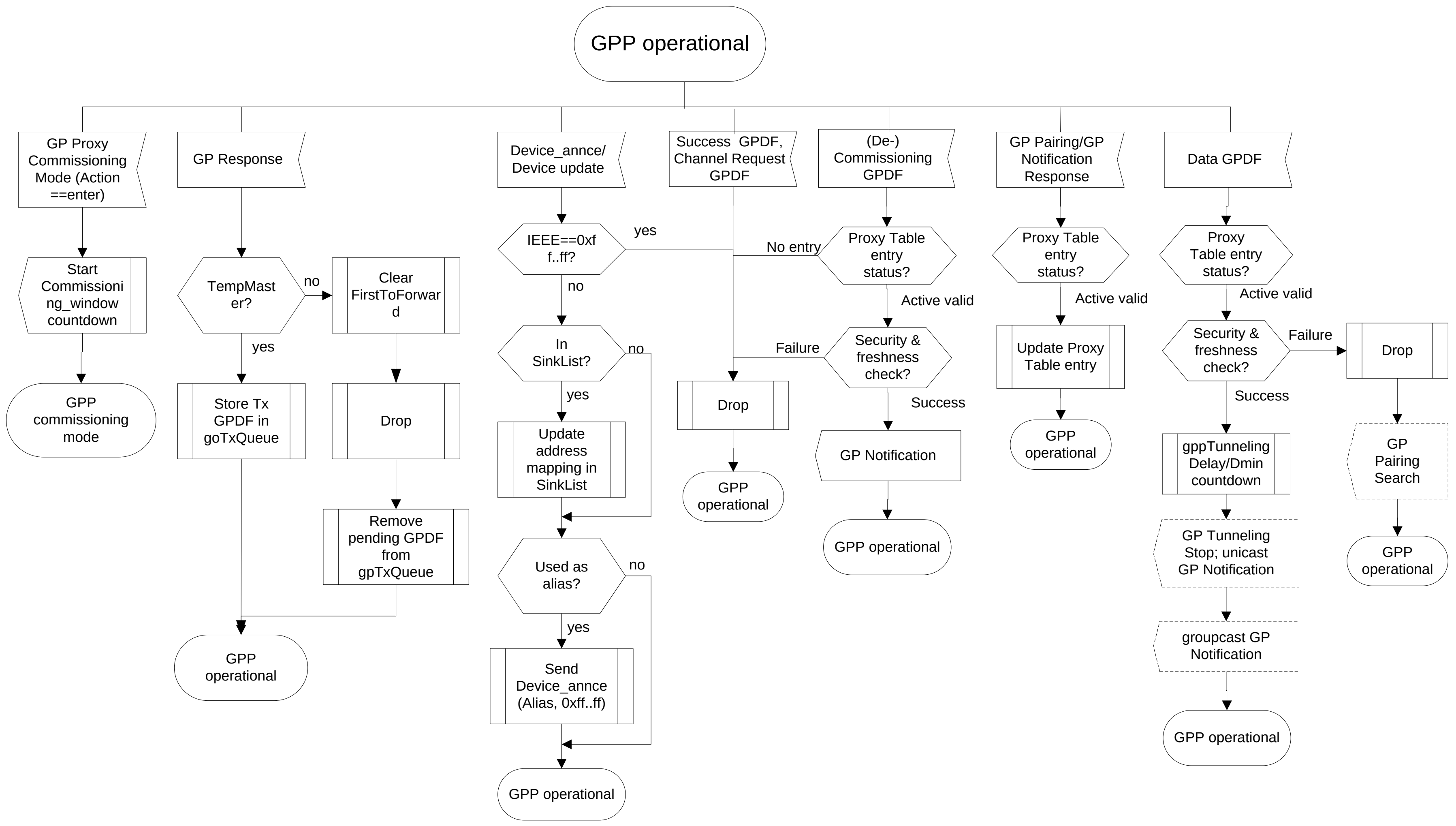


Figure 89 – Reconstruction of GPDF Frame Control fields by the sink

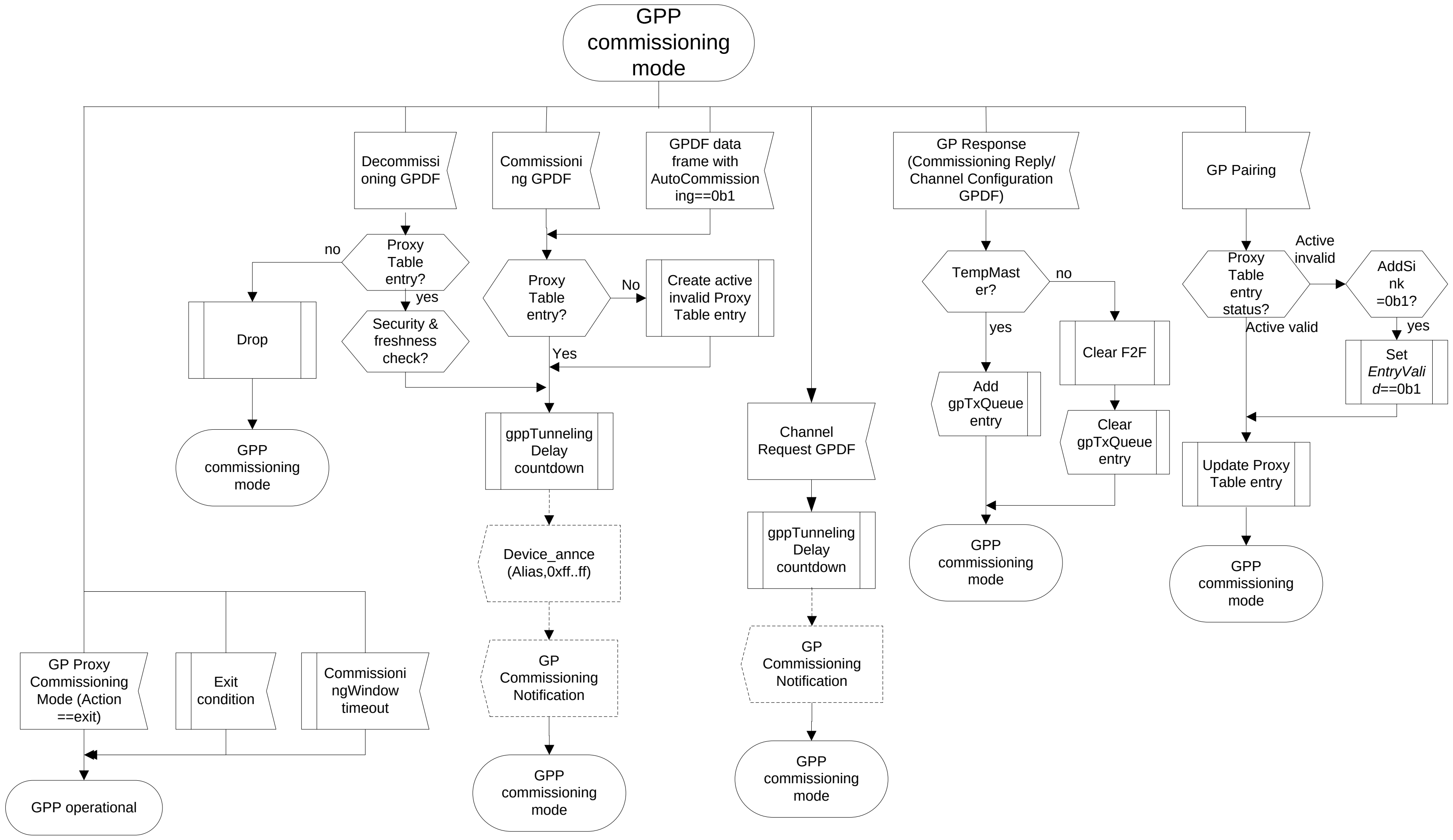
A.3.8 SDL diagrams for Green Power cluster operation

In this section, SDL diagrams are included, to provide high-level overview of the Green Power cluster operation. Please note, that this is high-level overview, and some detailed steps are not explicitly listed. Also, the application-specific behavior is on purpose not included.


GPP operational
GP Proxy Commissioning Mode (Action ==enter)
Start Commissioning_window countdown
GPP commissioning mode
GP Response
TempMaster?
no
yes
Clear FirstToForward
Drop
Remove pending GPDF from gpTxQueue
Store Tx GPDF in goTxQueue
GPP operational
Device_annce/ Device update
IEEE==0xff..ff?
yes
no
In SinkList?
no
yes
Update address mapping in SinkList
Used as alias?
no
yes
Send Device_annce (Alias, 0xff..ff)
GPP operational
Success GPDF, Channel Request GPDF
Drop
GPP operational
(De-) Commissioning GPDF
No entry
Proxy Table entry status?
Active valid
Failure
Security & freshness check?
Success
GP Notification
GPP operational
GP Pairing/GP Notification Response
Proxy Table entry status?
Active valid
Update Proxy Table entry
GPP operational
Data GPDF
Proxy Table entry status?
Active valid
Security & freshness check?
Failure
Drop
Success
gppTunneling Delay/Dmin countdown
GP Tunneling Stop; unicast GP Notification
groupcast GP Notification
GPP operational
GP Pairing Search
GPP operational

5869
5870 **Figure 90 – Proxy behavior in operational mode**

5871

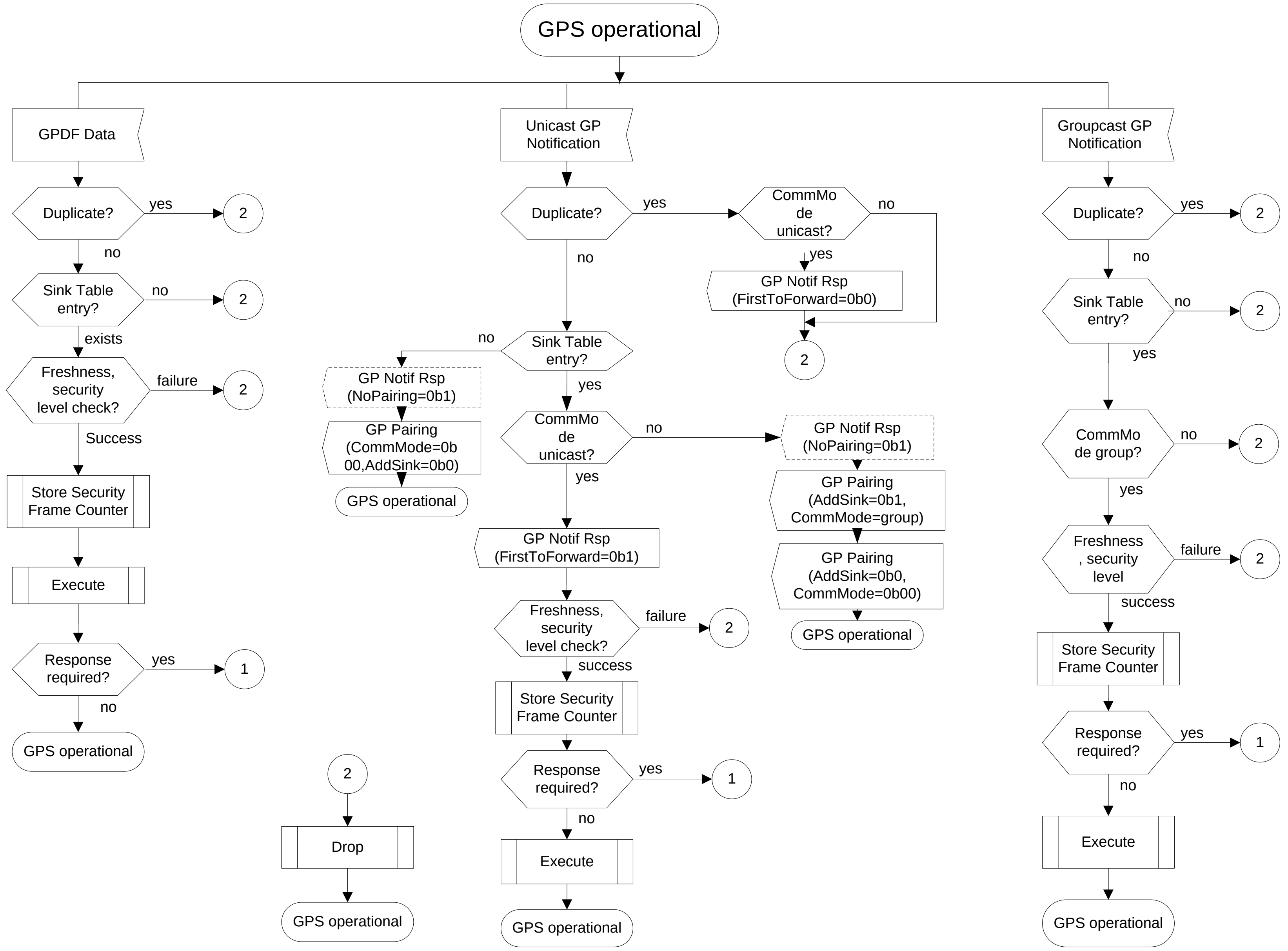


5872

5873 Figure 91 – Proxy behavior in commissioning mode

5874

5875



5876
5877 **Figure 92 – Sink behavior in operational mode (part 1)**

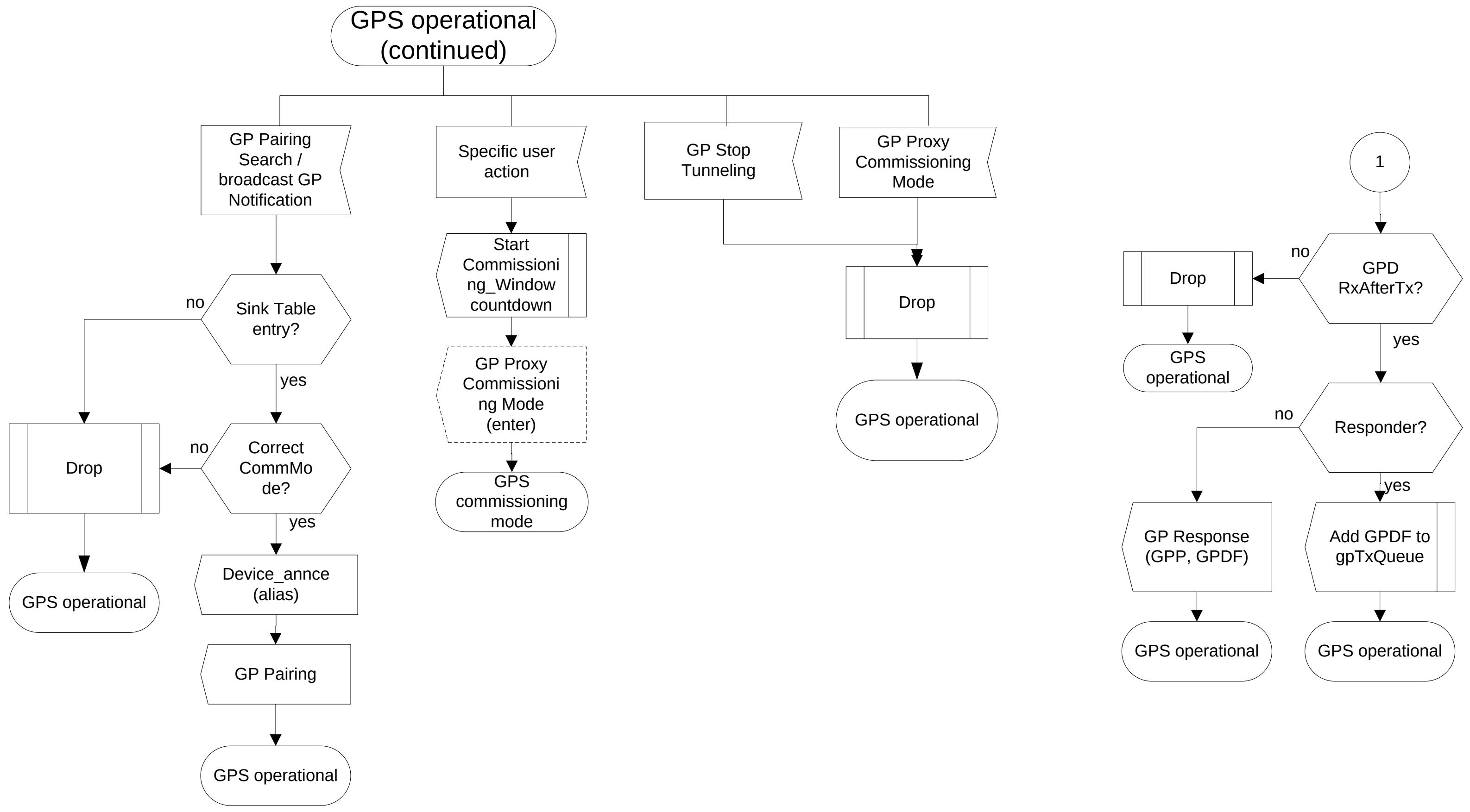
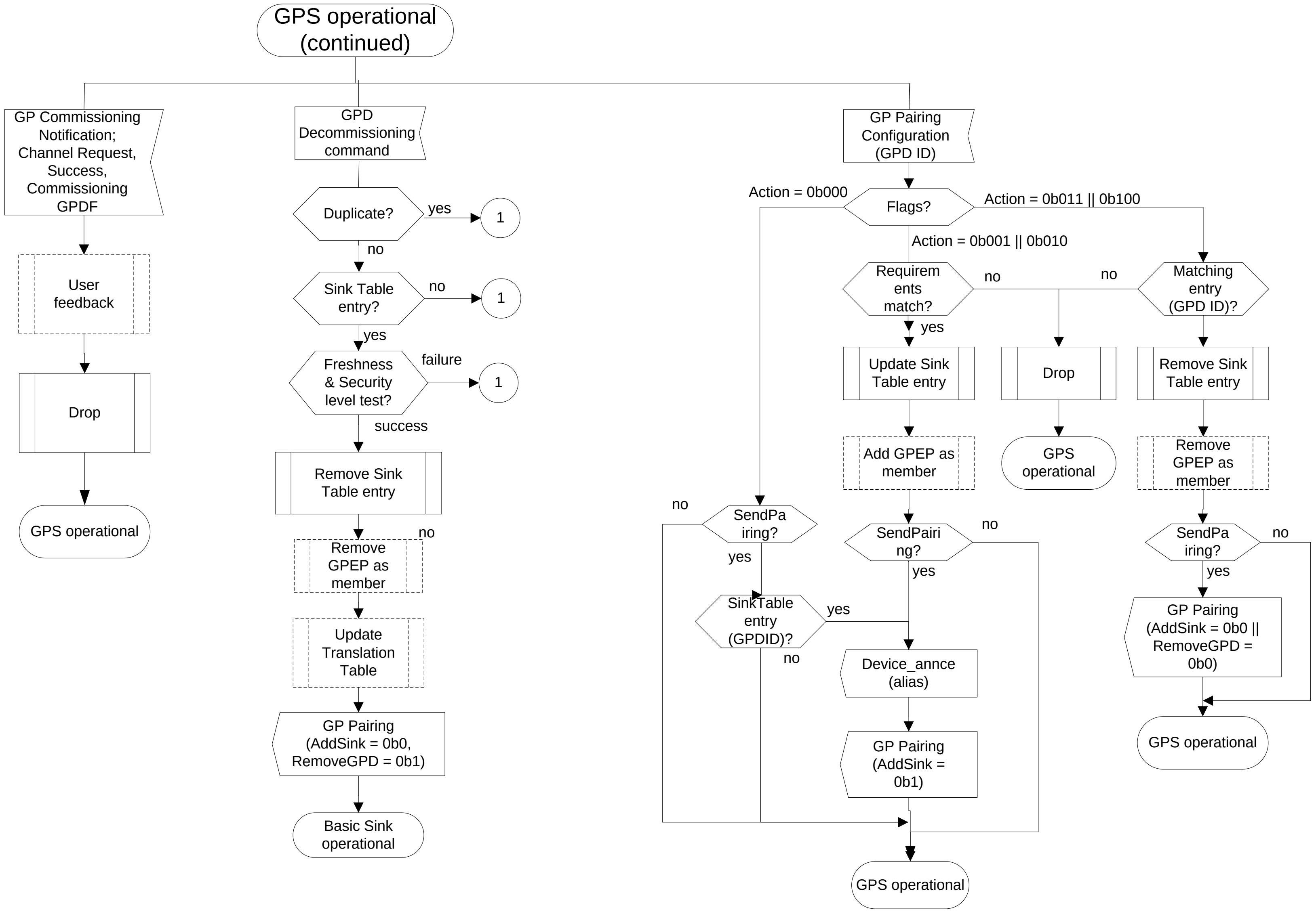


Figure 93 – Sink behavior in operational mode (part 2)



5882
5883 Figure 94 – Sink behavior in operational mode (part 3)
5884

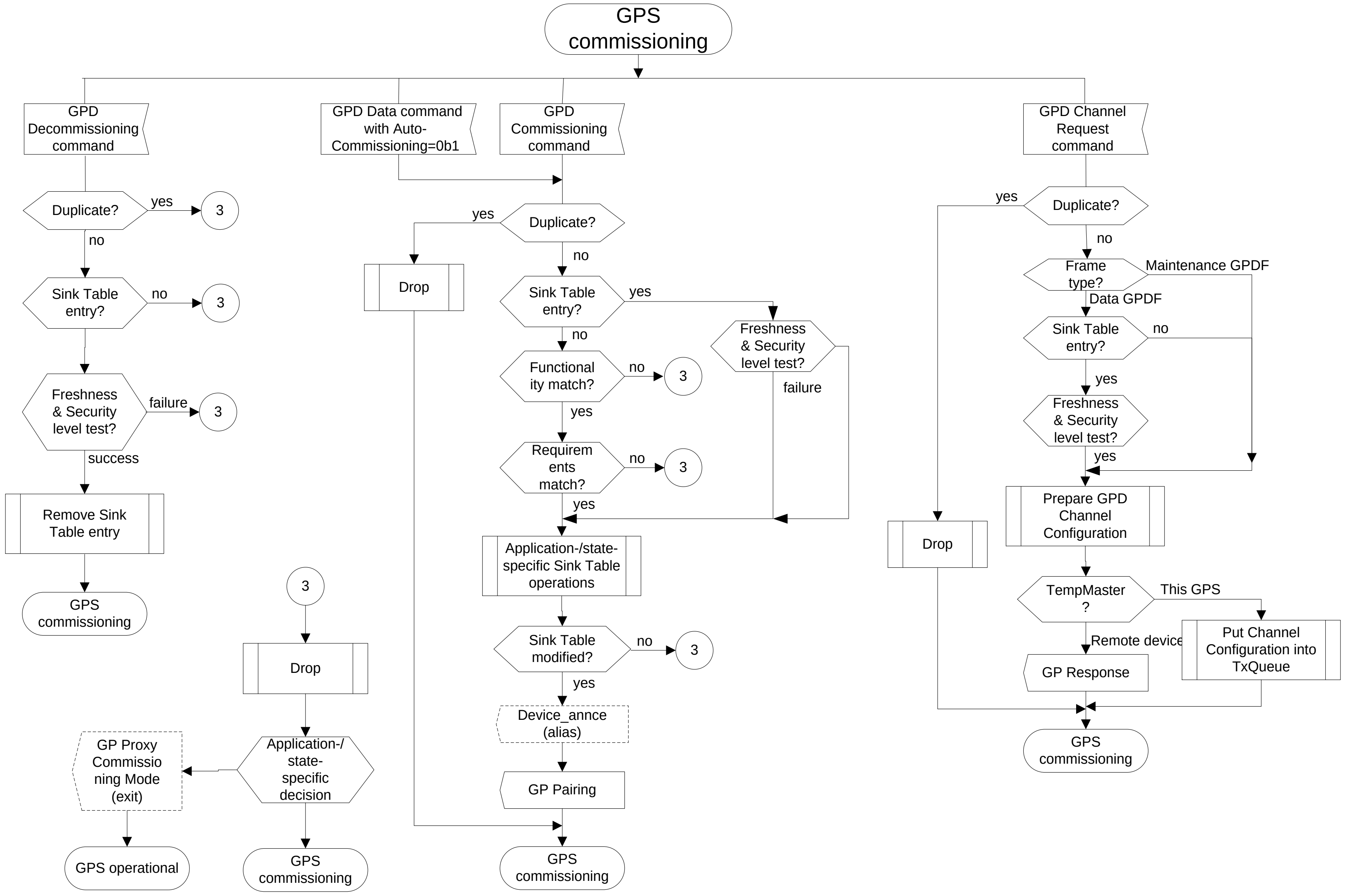

GPS commissioning
GPD Decommissioning command
GPD Data command with Auto-Commissioning=0b1
GPD Commissioning command
GPD Channel Request command
Duplicate?
yes
no
3
Sink Table entry?
Freshness & Security level test?
failure
success
Remove Sink Table entry
GPS commissioning
Drop
Functional ity match?
Requirem ents match?
Application-/state-specific Sink Table operations
Sink Table modified?
Device_annce (alias)
GP Pairing
Application-/ state-specific decision
GP Proxy Commissio ning Mode (exit)
GPS operational
Frame type?
Maintenance GPDF
Data GPDF
Prepare GPD Channel Configuration
TempMaster ?
This GPS
Remote device
Put Channel Configuration into TxQueue
GP Response

Figure 95 – Sink behavior in commissioning mode (part 1)

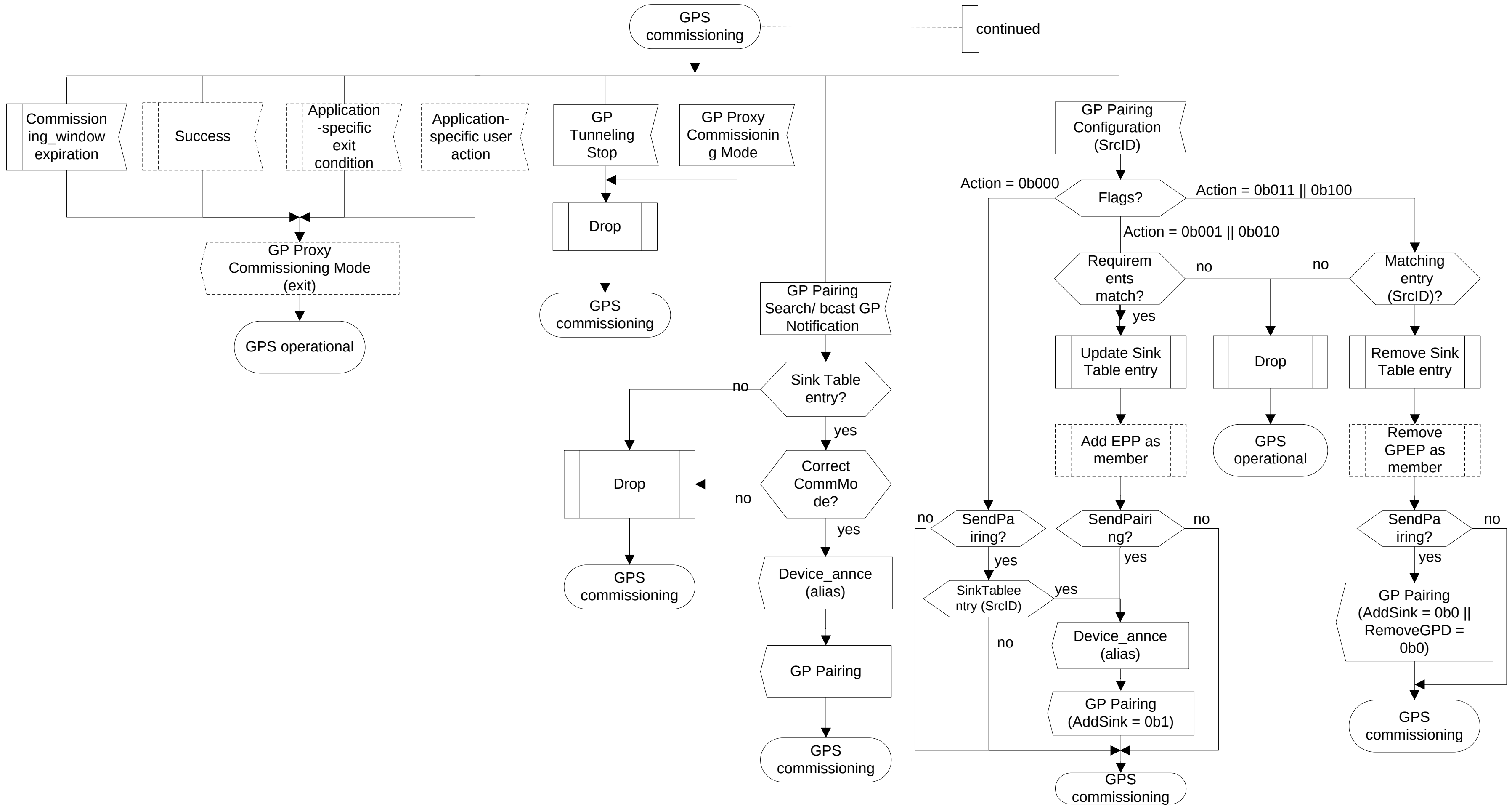
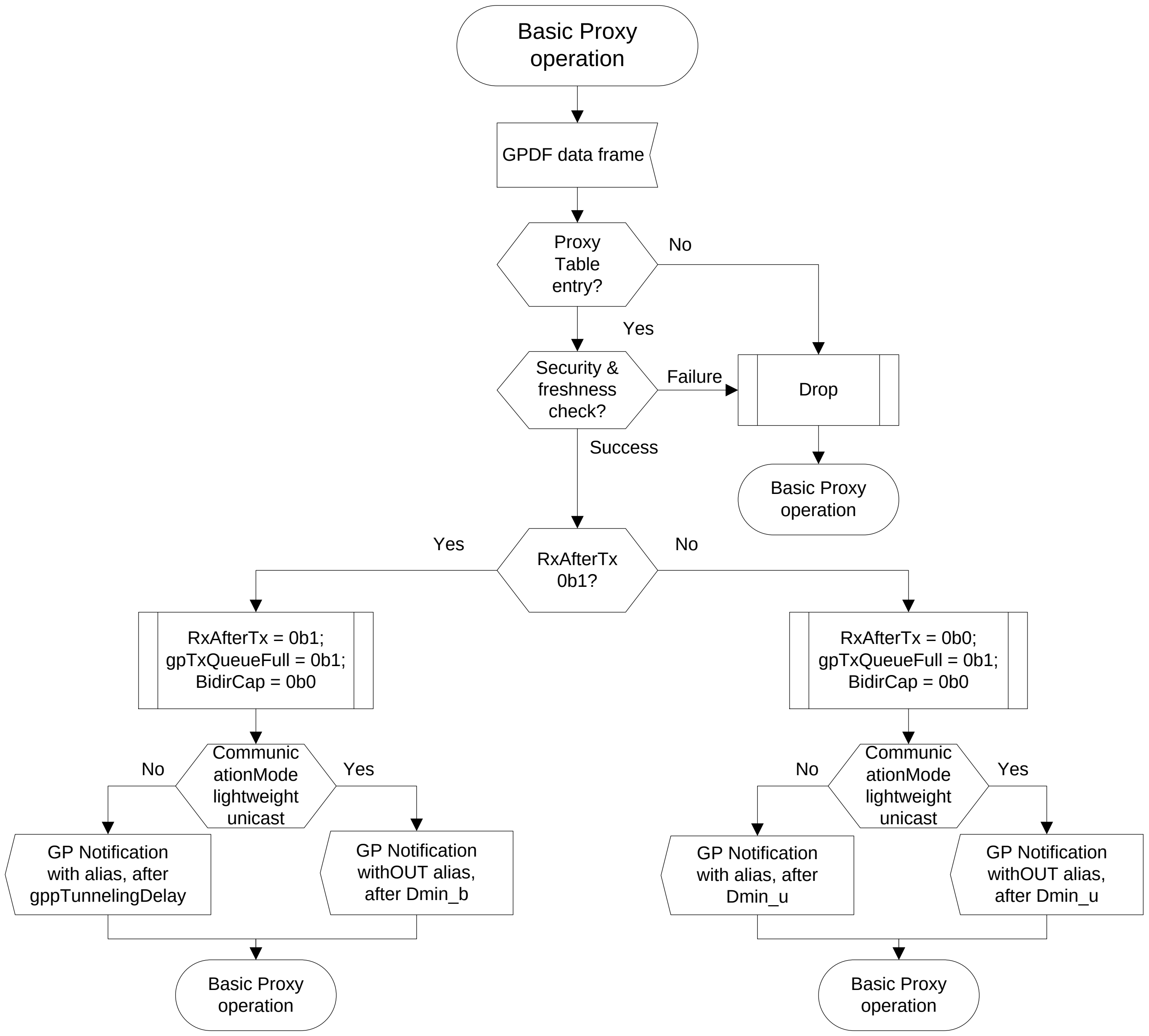
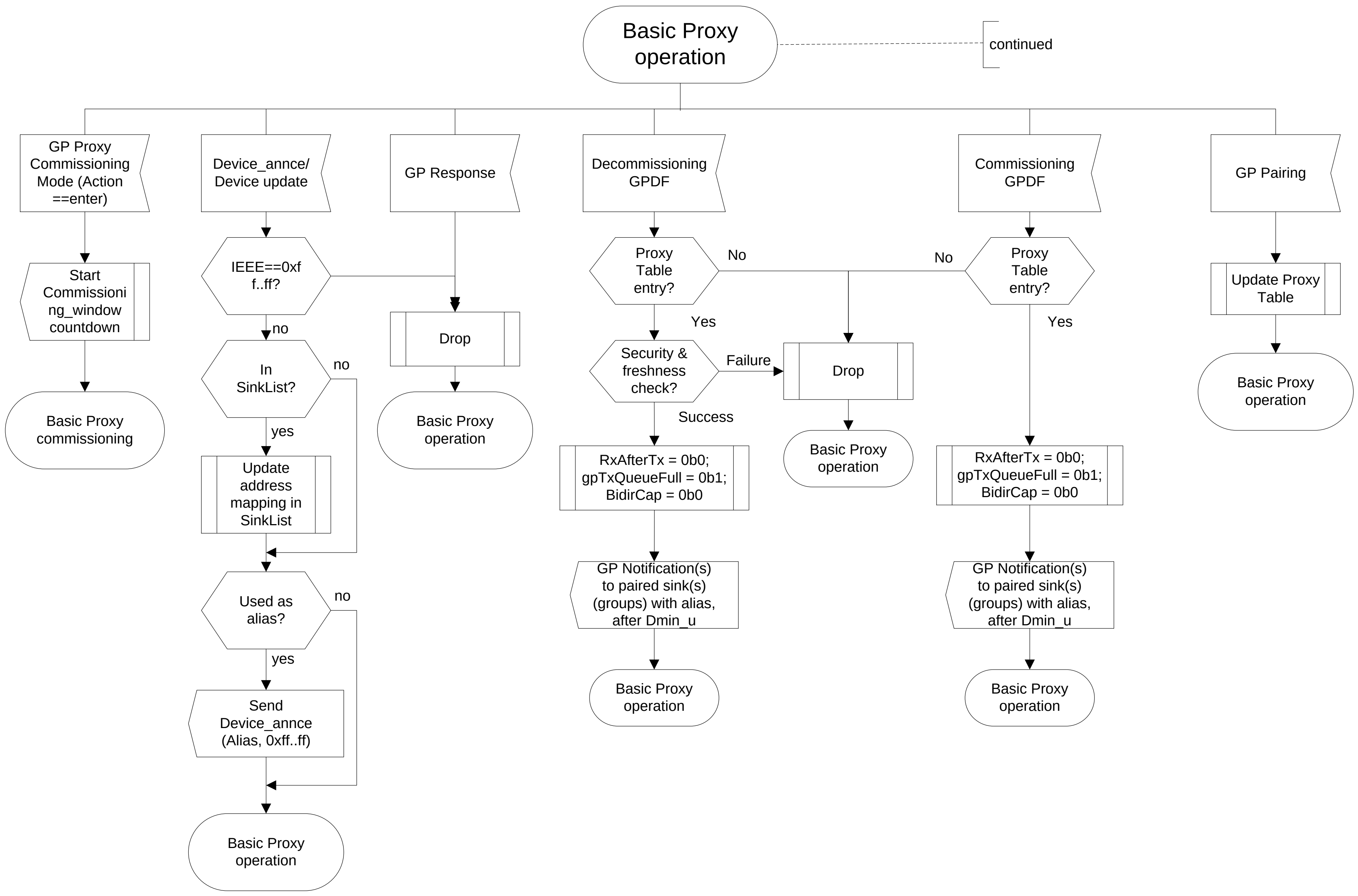


Figure 96 – Sink behavior in commissioning mode (part 2)

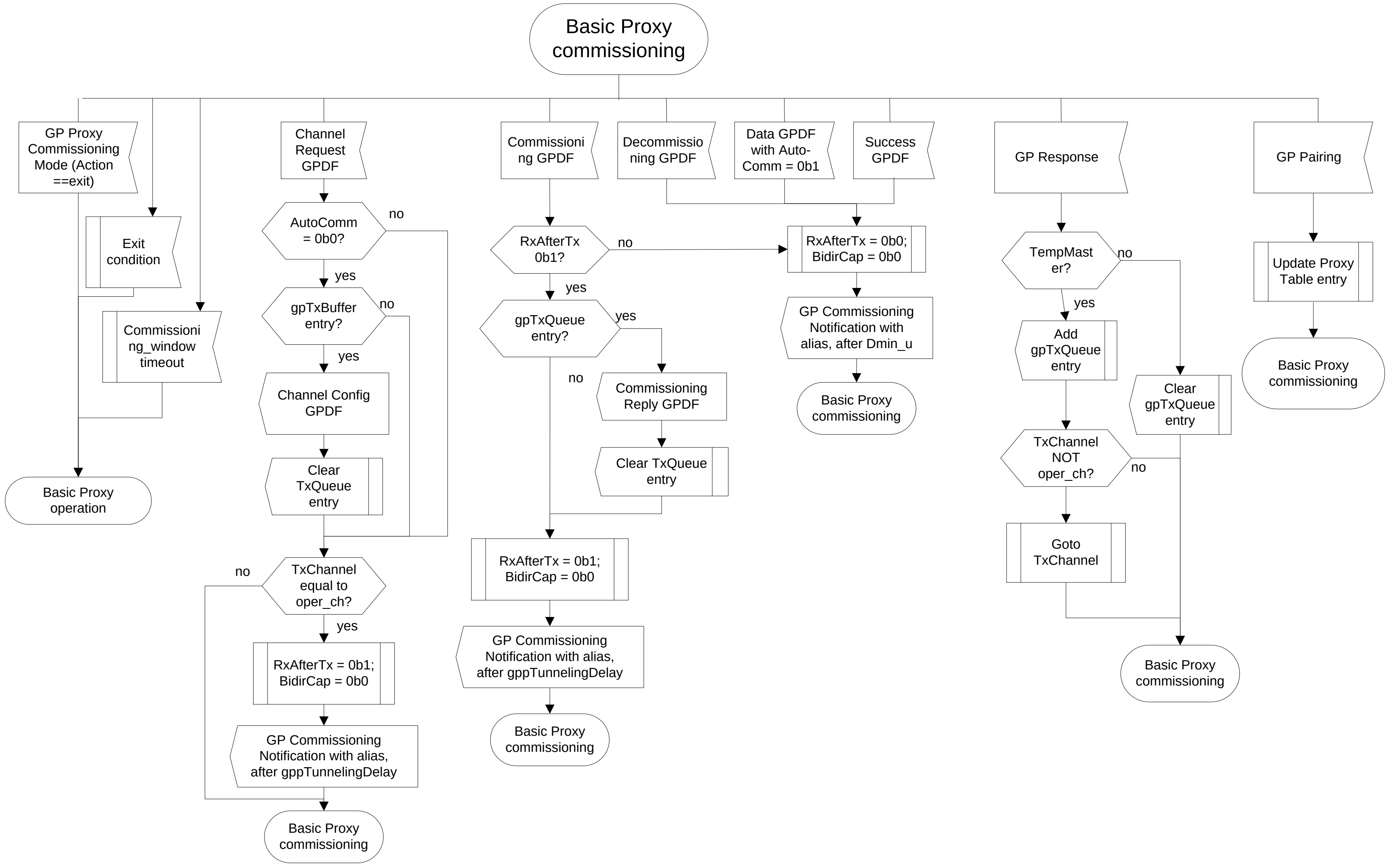
5898 ## A.3.8.1 GP Basic Proxy


Basic Proxy operation
GPDF data frame
Proxy Table entry?
No
Yes
Security & freshness check?
Failure
Drop
Success
Basic Proxy operation
Yes
RxAfterTx 0b1?
No
RxAfterTx = 0b1; gpTxQueueFull = 0b1; BidirCap = 0b0
RxAfterTx = 0b0; gpTxQueueFull = 0b1; BidirCap = 0b0
No
CommunicationMode lightweight unicast
Yes
No
CommunicationMode lightweight unicast
Yes
GP Notification with alias, after gppTunnelingDelay
GP Notification withOUT alias, after Dmin_b
GP Notification with alias, after Dmin_u
GP Notification withOUT alias, after Dmin_u
Basic Proxy operation
Basic Proxy operation

5899
5900 Figure 97 – GP Basic Proxy: behavior in operational mode (part 1)

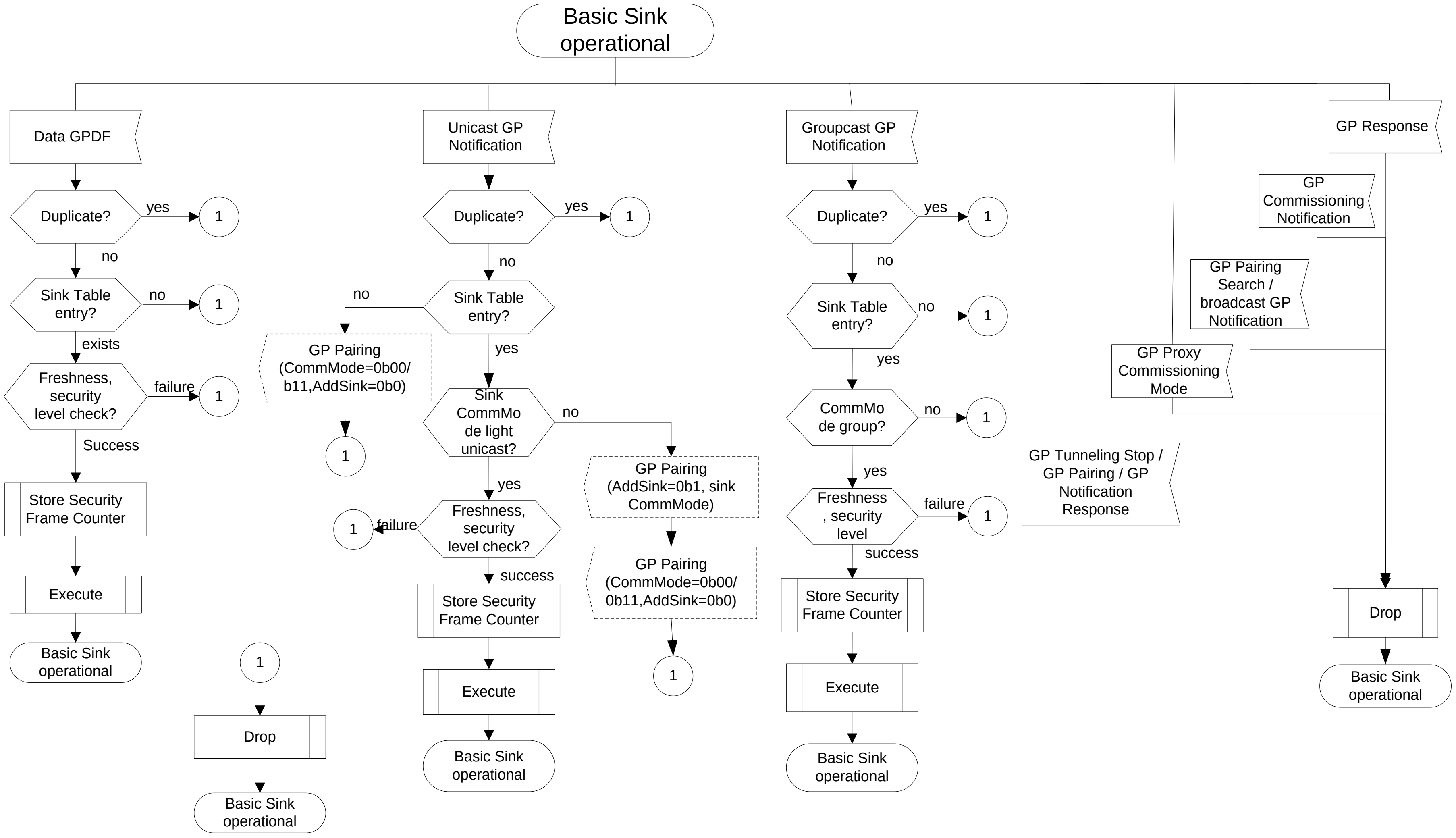

Basic Proxy operation
continued
GP Proxy Commissioning Mode (Action ==enter)
Start Commissioning_window countdown
Basic Proxy commissioning
Device_annce/ Device update
IEEE==0xff..ff?
no
In SinkList?
no
yes
Update address mapping in SinkList
Used as alias?
no
yes
Send Device_annce (Alias, 0xff..ff)
Basic Proxy operation
GP Response
Drop
Basic Proxy operation
Decommissioning GPDF
Proxy Table entry?
No
Yes
Security & freshness check?
Failure
Success
Drop
Basic Proxy operation
RxAfterTx = 0b0; gpTxQueueFull = 0b1; BidirCap = 0b0
GP Notification(s) to paired sink(s) (groups) with alias, after Dmin_u
Basic Proxy operation
Commissioning GPDF
Proxy Table entry?
No
Yes
RxAfterTx = 0b0; gpTxQueueFull = 0b1; BidirCap = 0b0
GP Notification(s) to paired sink(s) (groups) with alias, after Dmin_u
Basic Proxy operation
GP Pairing
Update Proxy Table
Basic Proxy operation

5901
5902 **Figure 98 – GP Basic Proxy: behavior in operational mode (part 2)**
5903


Basic Proxy commissioning
GP Proxy Commissioning Mode (Action ==exit)
Exit condition
Commissioning_window timeout
Basic Proxy operation
Channel Request GPDF
AutoComm = 0b0?
no
yes
gpTxBuffer entry?
no
yes
Channel Config GPDF
Clear TxQueue entry
TxChannel equal to oper_ch?
no
yes
RxAfterTx = 0b1; BidirCap = 0b0
GP Commissioning Notification with alias, after gppTunnelingDelay
Basic Proxy commissioning
Commissioning GPDF
RxAfterTx 0b1?
no
yes
gpTxQueue entry?
yes
no
Commissioning Reply GPDF
Clear TxQueue entry
RxAfterTx = 0b1; BidirCap = 0b0
GP Commissioning Notification with alias, after gppTunnelingDelay
Basic Proxy commissioning
Decommissioning GPDF
Data GPDF with Auto-Comm = 0b1
Success GPDF
RxAfterTx = 0b0; BidirCap = 0b0
GP Commissioning Notification with alias, after Dmin_u
Basic Proxy commissioning
GP Response
TempMaster?
no
yes
Add gpTxQueue entry
TxChannel NOT oper_ch?
no
Goto TxChannel
Clear gpTxQueue entry
Basic Proxy commissioning
GP Pairing
Update Proxy Table entry
Basic Proxy commissioning

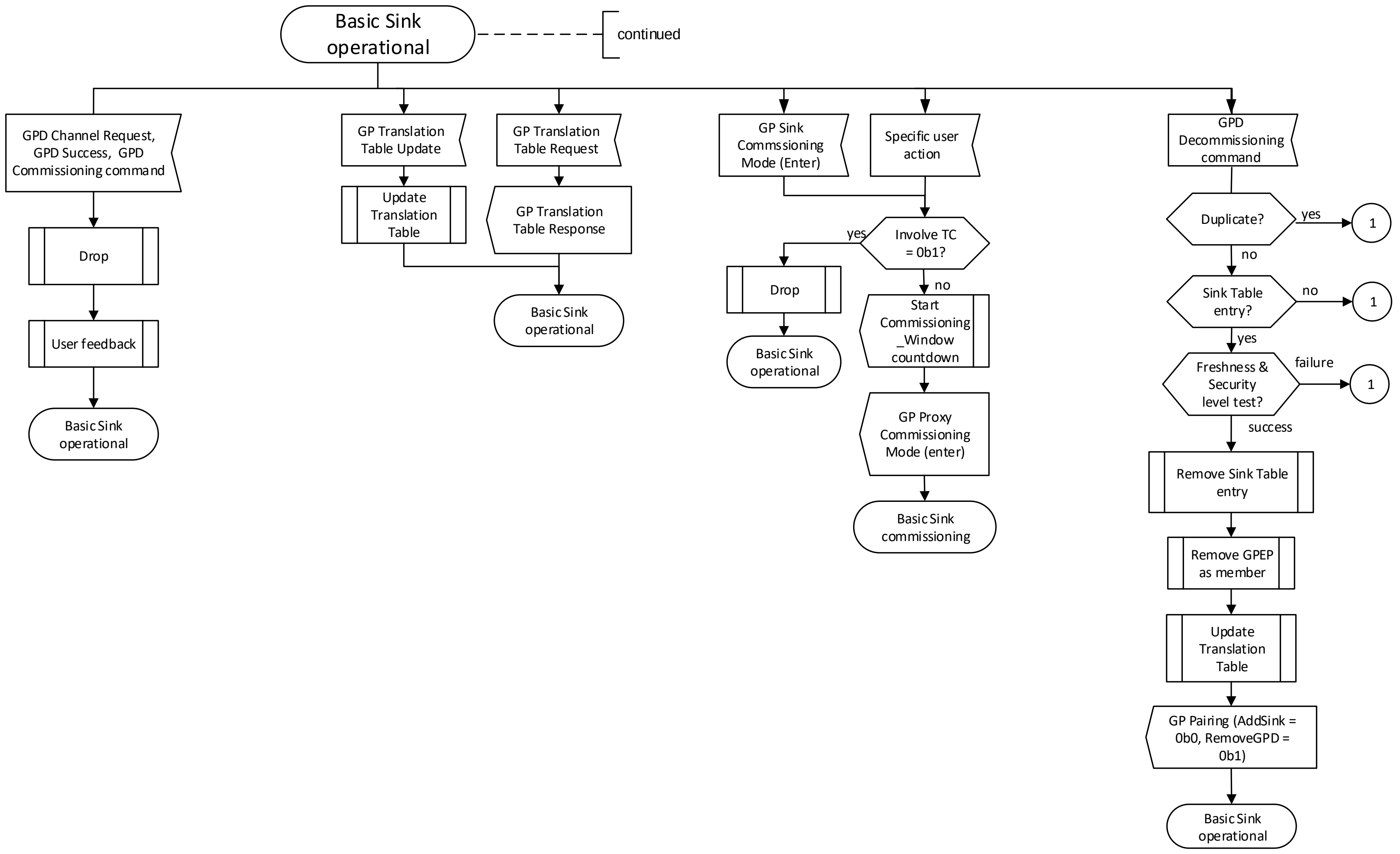
5904
5905
5906 **Figure 99 – GP Basic Proxy: behavior in commissioning mode**
5907

5908
A.3.8.2 Sink side of the GP Combo Basic


Basic Sink operational
Data GPDF
Duplicate?
yes
1
no
Sink Table entry?
no
exists
Freshness, security level check?
failure
Success
Store Security Frame Counter
Execute
Basic Sink operational
1
Drop
Basic Sink operational
Unicast GP Notification
Duplicate?
yes
no
Sink Table entry?
no
GP Pairing (CommMode=0b00/ b11,AddSink=0b0)
yes
Sink CommMode light unicast?
no
GP Pairing (AddSink=0b1, sink CommMode)
GP Pairing (CommMode=0b00/ 0b11,AddSink=0b0)
yes
Freshness, security level check?
failure
success
Store Security Frame Counter
Execute
Basic Sink operational
Groupcast GP Notification
Duplicate?
yes
no
Sink Table entry?
no
yes
CommMode group?
no
yes
Freshness, security level
failure
success
Store Security Frame Counter
Execute
Basic Sink operational
GP Response
GP Commissioning Notification
GP Pairing Search / broadcast GP Notification
GP Proxy Commissioning Mode
GP Tunneling Stop / GP Pairing / GP Notification Response
Drop
Basic Sink operational

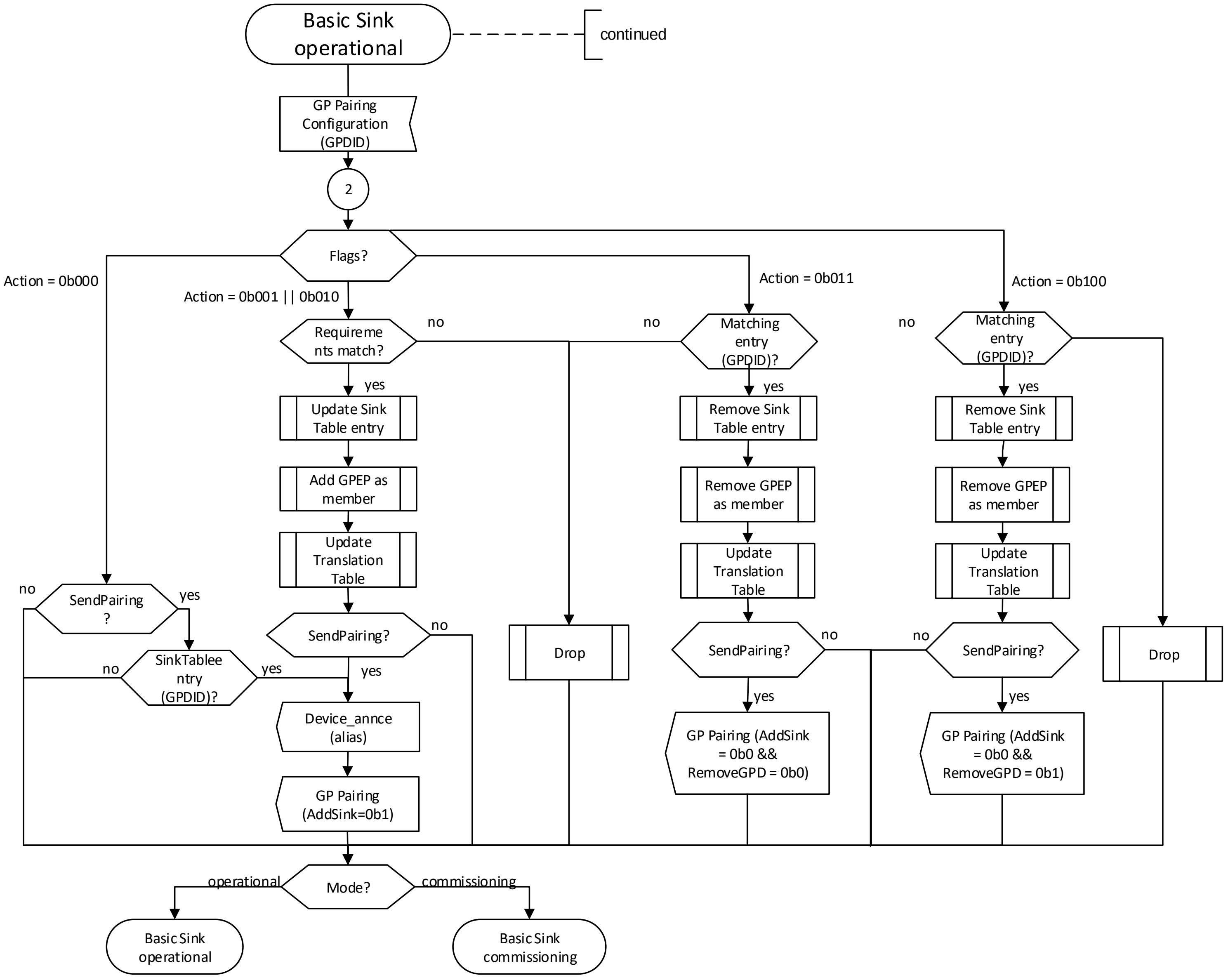
5909
5910 Figure 100 – GP Basic Sink: behavior in operational mode (part 1)

5911



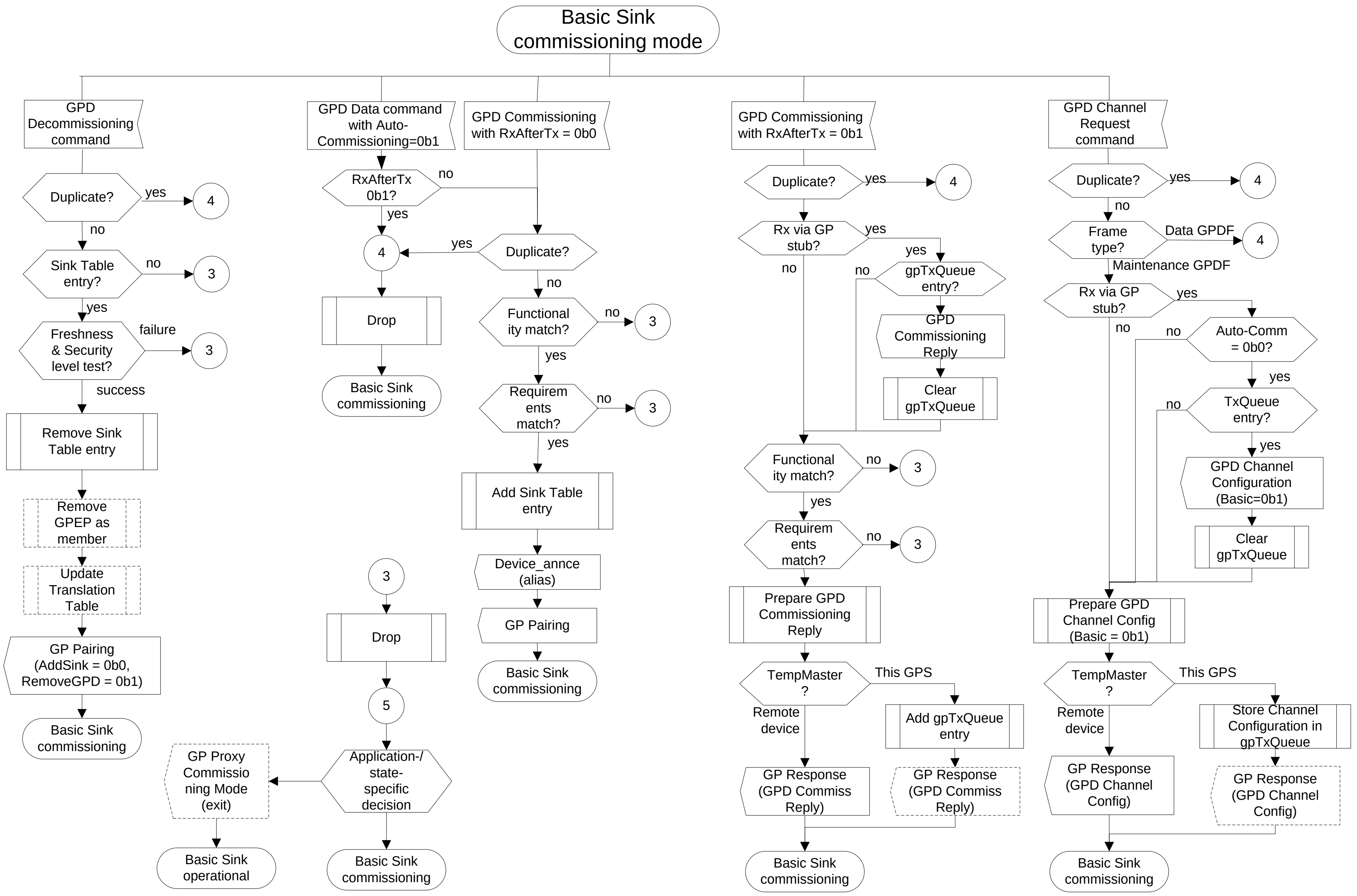
5912
5913 **Figure 101 – GP Basic Sink: behavior in operational mode (part 2)** [205]

[205] CCB #2323; Resolution added in 15-02014-011

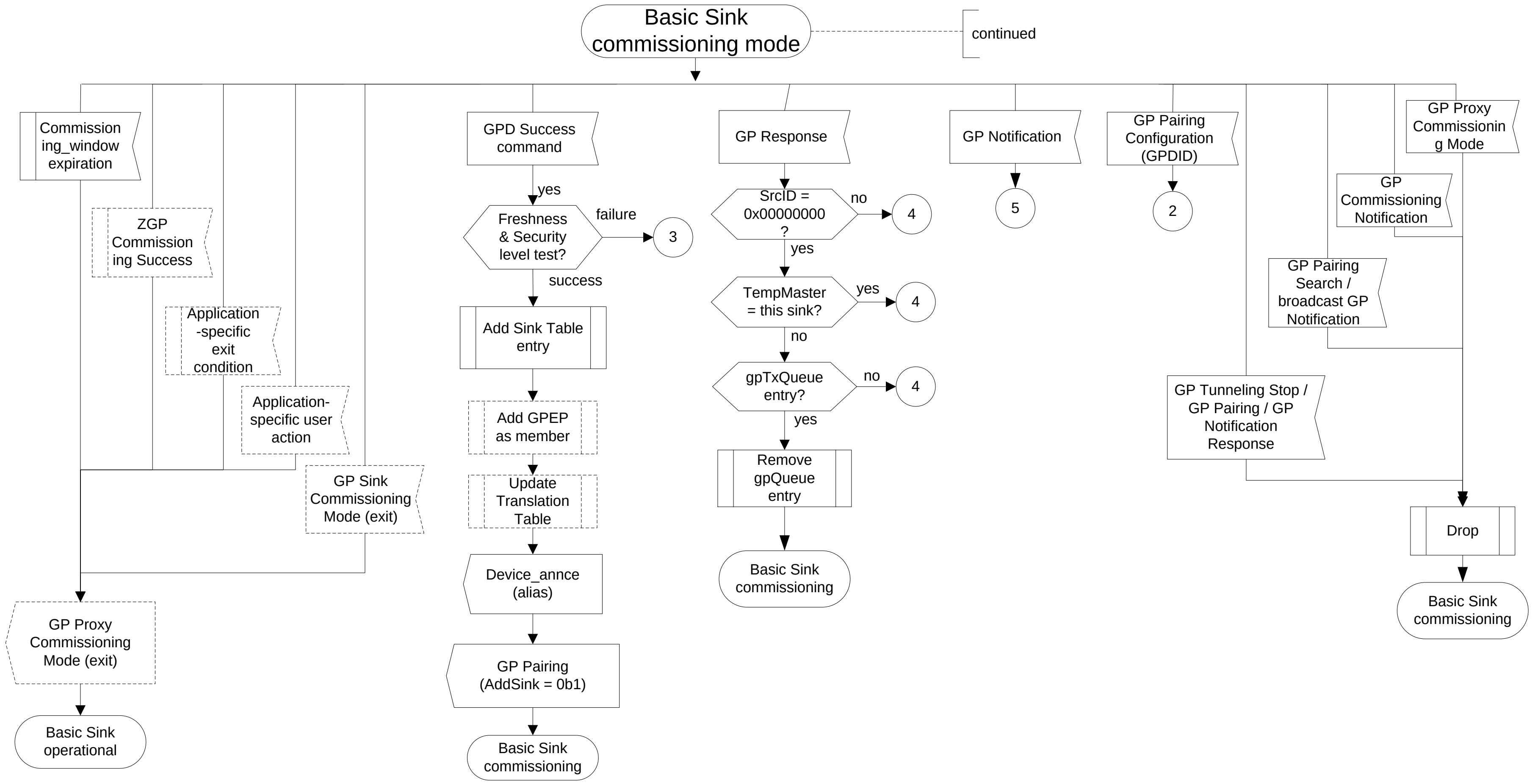


5914
5915 Figure 102 – GP Basic Sink: behavior in operational mode (part 3)[206]

[206] CCB #2323; Resolution added in 15-02014-011



5916
5917 **Figure 103 – GP Basic Sink: behavior in commissioning mode (part 1)**


Basic Sink commissioning mode
continued
Commissioning_window expiration
ZGP Commissioning Success
Application-specific exit condition
Application-specific user action
GP Sink Commissioning Mode (exit)
GP Proxy Commissioning Mode (exit)
Basic Sink operational
GPD Success command
yes
Freshness & Security level test?
failure
3
success
Add Sink Table entry
Add GPEP as member
Update Translation Table
Device_annce (alias)
GP Pairing (AddSink = 0b1)
Basic Sink commissioning
GP Response
SrcID = 0x00000000 ?
no
4
yes
TempMaster = this sink?
yes
4
no
gpTxQueue entry?
no
4
yes
Remove gpQueue entry
Basic Sink commissioning
GP Notification
5
GP Pairing Configuration (GPDID)
2
GP Proxy Commissioning Mode
GP Commissioning Notification
GP Pairing Search / broadcast GP Notification
GP Tunneling Stop / GP Pairing / GP Notification Response
Drop
Basic Sink commissioning

5918
5919 **Figure 104 – GP Basic Sink: behavior in commissioning mode (part 2)**

A.3.9 GP commissioning

The recommended GP commissioning procedure is described hereafter. The application profiles endorsing the Green Power feature MAY mandate it, or define another one, using the Green Power cluster commands.

It is left to the implementers of sink according to those methods, when to update the pairings in the Sink Table (add, modify or remove, dependent on different or the same user interaction, applications internal state, etc.), and when to exit commissioning mode (upon successful/failed pairing, timeout, user interaction, etc.). It is recommended, that the implementers make the sink behavior understandable to the user (e.g. via a user manual and/or appropriate user feedback). The profiles MAY define it further.

A.3.9.1 The procedure

1. **Enable commissioning on the sink**: the commissioning can be enabled on the sink in the following ways:
    a. The sink receives a GP Sink Commissioning Mode command with *Action* sub-field of the *Options* field set to 0b1.
    On reception of GP Sink Commissioning Mode command, if implemented, the sink SHALL behave as follows.
        i. In the current version of the specification, the sink SHALL first check if it needs to contact the Trust Centre, by checking the *Involve TC* sub-field of the *gpsSecurityLevel* attribute. If the *Involve TC* sub-field is set to 0b1, the sink SHALL NOT enter GP commissioning mode. If the *Involve TC* sub-field is set to 0b0, the sink SHALL act as follows.
        ii. If the *Action* sub-field of the *Options* field of the GP Sink Commissioning Mode command is set to 0b1, the sink SHALL enter the Green Power commissioning mode, for the application endpoint as indicated by the *Endpoint* field; value of 0xff indicates all active endpoints. If the *Involve proxies* sub-field of the *Options* field of the GP Sink Commissioning Mode command is set to 0b1 the sink SHALL, upon entering the commissioning mode, send the GP Proxy Commissioning Mode command, with the *Action* field set to 0b1 (i.e. Enter), the *Exit Mode* sub-field set according to the *gpsCommissioningExitMode* attribute, whereby the *CommissioningWindow* field MAY be included if required, and the *Channel present* sub-field set to 0b0; if the *Involve proxies* sub-field of the *Options* field of the GP Sink Commissioning Mode command is set to 0b0, the sink SHALL NOT send the GP Proxy Commissioning Mode command.
        If the *Action* sub-field of the *Options* field of the GP Sink Commissioning Mode command is set to 0b0, the sink SHALL exit the Green Power commissioning mode, for the application endpoint as indicated by the *Endpoint* field; value of 0xff indicates all active endpoints. If the *Involve proxies* sub-field of the *Options* field of the GP Sink Commissioning Mode command is set to 0b1 the sink SHALL, upon exiting the commissioning mode, send the GP Proxy Commissioning Mode command, with the *Action* field set to 0b0 (i.e. exit); if the *Involve proxies* sub-field of the *Options* field of the GP Sink Commissioning Mode command is set to 0b0, the sink SHALL NOT send the GP Proxy Commissioning Mode command
    b. The user enables commissioning on the sink via a vendor-specific action:
        i. In the current version of the specification, the sink SHALL first check if it needs to contact the Trust Centre, by checking the *Involve TC* sub-field of the *gpsSecurityLevel* attribute. If the *Involve TC* sub-field is set to 0b1, the sink SHALL NOT enter GP commissioning mode. If the *Involve TC* sub-field is set to 0b0, the sink SHALL act as follows.

ii. The sink enters commissioning mode

iii. Optionally (depending on the vendor-specific requirements) the sink sends on the operational channel a GP Proxy Commissioning Mode command (with *Action* sub-field of the *Options* field set to 0b1 = enter; indicating the *Exit mode*, indicating the required communication mode by setting or clearing the *Unicast communication* sub-field, optionally overriding the duration of the default *gppCommissioningWindow*, e.g. to 0xffff by setting the *Options* sub-fields accordingly).
Note: Hereafter we use the term multi-hop commissioning to indicate that this option is applied, and the term proximity commissioning to indicate that this option is not applied. In the proximity commissioning, the commissioned sink and the GPD are the only involved parties. If multi-hop commissioning is enabled AND the sink supports direct communication, and the sink is in direct range of the GPD, then the sink SHALL also consider itself as a candidate TempMaster; i.e. enabling multi-hop commissioning SHALL also enable the sink for proximity commissioning, if supported.

2. **Proxies enter commissioning mode**: The proxies receiving a GP Proxy Commissioning Mode (*Action*=enter) command on the operational channel (if sent) in operational mode SHALL store the address of the originator, start the *CommissioningWindow*/*gppCommissioningWindow* timeout (see sec. A.3.3.2.5/A.3.6.3.2) to exit commissioning mode in case of no pairing/no explicit exit command, and enter commissioning mode on the operational channel.
While in commissioning mode, the proxies SHALL only accept GP Proxy Commissioning Mode commands from the device that originally put them in commissioning mode, and SHALL silently drop GP Proxy Commissioning Mode commands from other devices.
If the *Unicast communication* sub-field of the *Options* field was set to 0b0, the receiving proxies SHALL send the GP Commissioning Notification commands in broadcast; if set to 0b1, they SHALL send the GP Commissioning Notification commands in unicast to the originator of the GP Proxy Commissioning Mode command.
While in commissioning mode, the proxies SHALL process all other commissioning-related commands (e.g. GP Pairing), from all senders.

3. **GPD commissioning state machine**: The user triggers the commissioning action (and repeats it, if required, depending on the energy budget of the GPD) on the GPD (and *Endpoint*, specific or 0xff, if *ApplicationID* = 0b010) **until success feedback or failure feedback is provided by the commissioning sink**.

[207]If **subsequent commissioning** is triggered on the GPD, the GPD SHALL proceed as defined in sec. A.1.7.3.2.

Note: The user SHOULD NOT push too quickly, in order to allow the system to process the messages and provide the success feedback, if any. E.g. 1 push a second.
If the GPD capable of bidirectional automatically advances between the successive commissioning steps, it also SHOULD NOT do it too quickly, in order to allow the infrastructure devices involved to perform the necessary steps. It is recommended to have at least 200ms delay between two consecutive commissioning steps comprising the transmission of a series of GPD Channel Request commands or a GPD Commissioning command with RxAfterTx *sub-field of the* Extended NWK

[207] Generic switch commissioning guidelines, Zigbee document 16-02604-004
Dec 2016 SVE comment: https://workspace.zigbee.org/kws/groups/PRO_GP/comments/view_comment?comment_id=1012

Frame Control *field set to 0b1). For consecutive commissioning steps comprising the transmission of a GPD Success command or a GPD Commissioning command with* RxAfterTx *sub-field of the* Extended NWK Frame Control *field set to 0b0, it is sufficient to have at least 50ms delay.*
Note2: the internal commissioning state of the GPD capable of setting RxAfterTx *during commissioning is assumed to be represented by two internal state variables:* ToggleChannel *variable and* ParametersStored *variable.*

a. If the GPD is in commissioning mode AND *BidirectionalCommissioning* variable is TRUE AND its internal *ToggleChannel* variable is TRUE,

   i. the GPD sends a GPD Channel Request command in a GPDF on the supported number of channels per attempt; the Channel Request GPDF SHALL be sent using the Maintenance frame type, and unprotected [208](even for subsequent commissioning attempts); the *Auto-Commissioning* sub-field of the *NWK Frame Control* field SHALL be set to 0b0 in a GPD Channel Request frame immediately followed by a reception window. If multiple GPD Channel Request frames are sent per reception window, the *Auto-Commissioning* sub-field of all the GPD Channel Request frames immediately followed by another transmission of GPD Channel Request SHALL be set to 0b1. The *MAC Sequence number* value for each transmission of Channel Request GPDF SHOULD be different; if *SecurityLevelCapabilities* = 0b00 and *MACsequenceNumberCapability* = 0b1, the *MAC sequence number* SHALL be incremental.
   Note: the number of channels the GPD can send the channel request on for a single commissioning attempt is defined by the energy budget of each particular GPD. The GPD vendor needs to make sure, that after the transmission (of the series), the GPD is still able to receive the Channel Configuration GPDF and non-volatilely store the number of the operational channel, as well as the state information.

   ii. *gpdRxOffset* ms after the start of the transmission of the (first) Channel Request with *Auto-Commissioning* = 0b1 sent on the Rx channel for this attempt, the GPD enters Rx mode on this channel for at least the duration of *gpdMinRxWindow*.

   iii. **GOTO step 4 (for Multi-hop commissioning) or step 5 (for proximity commissioning)**.

b. If the GPD is in commissioning mode AND the GPD does NOT support the GPD Compact Attribute Reporting command AND *BidirectionalCommissioning* variable is TRUE AND its internal *ToggleChannel* variable is FALSE AND its *ParametersStored* variable is FALSE as well,

   i. the GPD sends a Commissioning GPDF on the operational channel with the *Auto-Commissioning* sub-field of the *NWK Frame Control* field set to 0b0, *RxAfterTx*=0b1; the security related fields are set as defined in A.3.9.2. Also, the GPD sets the appropriate fields of the (*Extended) Options* field to request the further configurations parameter it needs. [209]In the current version of the specification, the Commissioning GPDF SHALL always be sent unprotected, including subsequent commissioning.
   If *GPDoutgoingCounter* field is present in the payload of the GPD Commissioning command (and it SHALL if *SecurityLevelCapabilities* sub-field of the *Extended Options* field is set to 0b10 or 0b11), the value it carries SHALL be incremented for every transmission of a Commissioning GPFS.
   The *MAC Sequence number* value for each transmission of Commissioning GPDF SHOULD be different; if *SecurityLevelCapabilities* = 0b00 and *MACsequenceNumberCapability* = 0b1, the *MAC sequence number* SHALL be incremental; it MAY but is not required to be

[208] Generic switch commissioning guidelines, Zigbee document 16-02604-004
[209] Dec 2016 SVE comment: https://workspace.zigbee.org/kws/groups/PRO_GP/comments/view_comment?comment_id=1012

aligned with the *GPDoutgoingCounter* field in the payload of the GPD Commissioning command.

ii. *gpdRxOffset* ms after the start of the transmission of the first Commissioning GPDF in GPFS, the GPD enters Rx mode on the operational channel for at least the duration of *gpdMinRxWindow*.

iii. **GOTO step 4 (for Multi-hop commissioning) or step 5 (for proximity commissioning)**.

c. If the GPD is in commissioning mode AND the GPD does NOT support the GPD Compact Attribute Reporting command AND *BidirectionalCommissioning* variable is TRUE AND its internal *ToggleChannel* variable is FALSE AND its *ParametersStored* variable is TRUE, the GPD sends a Success GPDF on the operational channel with the *Auto-Commissioning* sub-field of the *NWK Frame Control* field set to 0b0; if the *Extended NWK Frame Control* field is present, then the *RxAfterTx*=0b0.
If security is to be used by this GPD, the Success GPDF SHALL be appropriately secured; the value of the *Security frame counter* field in the NWK header of the Success GPDF SHALL be higher than the last used value of the *GPDoutgoingCounter* field in the payload of the GPD Commissioning command. The *MAC Sequence number* SHOULD be different than that in the last Commissioning GPDF; if *SecurityLevelCapabilities* = 0b00 and *MACsequenceNumberCapability* = 0b1, the *MAC sequence number* SHALL be incremental; it MAY but is not required to be aligned with the *Security frame counter* field.
Note: If *gpdSecurityLevel* = 0b11, the Success GPDF SHALL be secured *SecurityLevel* = 0b11.
If the GPD automatically progresses to transmission of Success GPDF (without a separate user interaction/user trigger), then the Success GPDF SHALL be sent at least 50ms after the successful reception of GPD Commissioning Reply command.
If more than one Success GPFS is sent (as is recommended to increase the probability of reception), and if *gpdSecurityLevel* is set to 0b10 or 0b11, the security frame counter SHALL be incremented for every transmission of a Success GPFS.
GOTO step 4 (for Multi-hop commissioning) or step 5 (for proximity commissioning).

d. If the GPD is in commissioning mode AND the GPD does NOT support the GPD Compact Attribute Reporting command AND *BidirectionalCommissioning* variable is FALSE, and the GPD is capable of sending Commissioning GPDFs, the GPD sends a Commissioning GPDF on one channel, with the *Auto-Commissioning* sub-field of the *NWK Frame Control* field set to 0b0 and *RxAfterTx*=0b0, and the security related fields are set as defined in A.3.9.2. Also, the GPD sets the sub-fields of the *Options* field appropriately.
If *GPDoutgoingCounter* field is present in the payload of the GPD Commissioning command (and it SHALL if *SecurityLevelCapabilities* sub-field of the *Extended Options* field is set to 0b10 or 0b11), the value it carries SHALL be incremented for every transmission of a Commissioning GPFS.
The *MAC Sequence number* value for each transmission of Commissioning GPDF SHOULD be different; it MAY but is not required to be aligned in any way with the *GPDoutgoingCounter* field in the payload of the GPD Commissioning command.
The GPD SHOULD start with the last memorized channel.
GOTO step 4 (for Multi-hop commissioning) or step 5 (for proximity commissioning).

e. If the GPD is in commissioning mode AND *BidirectionalCommissioning* variable is FALSE and the GPD is not capable of sending Commissioning GPDF, i.e. Data GPDF with *Auto-Commissioning* set to 0b1 is sent, *RxAfterTx* sub-field, if present, is set to 0b0, there is probably a special action for the user to set the channel on the GPD (e.g. DIP switches).
GOTO step 12 (for Multi-hop commissioning) or step 13 (for proximity commissioning).

According to the current version of the specification, only GPD that support *gpdSecurityLevel* = 0b10 or higher AND support TC-LK protection (as indicated by the *GPDkeyEncryption* sub-field of the *Extended Options* field of the GPD Commissioning command) of the GPD key, if exchanged over the air, can be certified.

f. If the GPD is in commissioning mode AND the GPD is capable of sending Commissioning GPDFs AND the GPD supports the GPD Compact Attribute Reporting command AND *BidirectionalCommissioning* variable is FALSE,
the GPD sends a Commissioning GPDF on one channel, formatted as specified in step 3.d. above, but with the *GPD Application Description command follows* sub-field of the *Application Information* field is set to 0b1.
Immediately after transmitting the Commissioning GPDF, the GPD SHALL send, on the same channel, (all) the GPD Application Description command(s), unprotected, and with *RxAfterTx* set to 0b0.
Note: depending on the GPD's energy budget, the transmission of the GPD Application Description command(s) may require an additional commissioning action; then, the GPD SHALL store the information about the Report identifier *values already sent in Application Description GPDFs following the current Commissioning GPDF.*
GOTO step 4 (for Multi-hop commissioning) or step 5 (for proximity commissioning).

g. If the GPD is in commissioning mode AND the GPD is capable of sending Commissioning GPDFs AND the GPD supports the GPD Compact Attribute Reporting command AND *BidirectionalCommissioning* variable is TRUE AND its internal *ToggleChannel* variable is FALSE AND its *ParametersStored* variable is FALSE as well,

   i. the GPD sends a Commissioning GPDF on one channel, formatted as specified in step 3.b.i. above, but with the *RxAfterTx* sub-field of the *Extended Network Frame Control* field set to 0b0 and the *GPD Application Description command follows* sub-field of the *Application Information* field is set to 0b1.
   Immediately after the Commissioning GPDF, the GPD SHALL send, on the same channel, (all) the GPD Application Description command(s), unprotected; only the last GPD Application Description command following one particular Commissioning GPDF (i.e. the Application Description GPDF carrying the highest *Report identifier* supported by this GPD) SHALL have *RxAfterTx* set to 0b1; all preceding Application Description GPDF SHALL have *RxAfterTx* set to 0b0.
   Note: depending on the GPD's energy budget, the transmission of the GPD Application Description command(s) may require an additional commissioning action.

   ii. *gpdRxOffset* ms after the start of the transmission of the first [210]Application Description GPDF with *RxAfterTx* set to 0b1 in GPFS, the GPD enters Rx mode on the operational channel for at least the duration of *gpdMinRxWindow*.

   iii. **GOTO step 4 (for Multi-hop commissioning) or step 5 (for proximity commissioning).**

4. **Proxy commissioning state machine:** proxy in radio range of the commissioning GPD receives on the operational channel (unless explicitly stated otherwise):
   a. Channel Request GPDF – **GOTO step 6**;
   b. Channel Request GPDF on the *TransmitChannel* – **GOTO step 9**;
   c. Channel Configuration GPDF – **GOTO step 11**;

[210] Comment #778 from GP multi-sensor v0.7 letter ballot

d. Commissioning GPDF or Data GPDF with *Auto-Commissioning* set to 0b1 [211]or Application Description GPDF – **GOTO step 12**;
e. Commissioning Reply GPDF – **GOTO step 16**;
f. Success GPDF – **GOTO step 17**.

5. **Sink commissioning state machine**: the sink receives – either directly, if in radio range of the commissioning GPD, or in GP Commissioning Notification – on the operational channel (unless explicitly stated otherwise):
   a. Channel Request GPDF – **GOTO step 7**;
   b. Channel Request GPDF on the *TransmitChannel* – **GOTO step 9**;
   c. Channel Configuration GPDF – **GOTO step 11**;
   d. Commissioning GPDF or, if supported, Data GPDF with *Auto-Commissioning* set to 0b1 [212]or Application Description GPDF – **GOTO step 13**;
   e. Commissioning Reply GPDF – **GOTO step 16**;
   f. Success GPDF – **GOTO step 18**.
   [213]Note: the commissioning information allowing the sink to distinguish unidirectional and bidirectional commissioning procedure being currently performed must be kept for the duration of the procedure, since in case of bidirectional commissioning of a GPD capable of compact attribute reporting not all of the commissioning commands have the *RxAfterTx* sub-field of the *Extended NWK Frame Control* set to 0b1 (specifically: only the last Application Description GPDF will have the *RxAfterTx* = 0b1, the Commissioning GPDF and other Application Description GPDFs, if any, will have *RxAfterTx* = 0b0).

<u>**In-band channel determination part**</u>

6. **Proxy receives Channel Request GPDF**: The proxies in radio range of the GPD receiving the Channel Request GPDF on the operational channel,
   a. If they are NOT in commissioning mode: silently drop the Channel Request.
   [214]If the proxy received the GPDF in commissioning mode and the *Frame Type* sub-field of the *NWK Frame Control* field was set to 0b01, the *Auto-Commissioning* sub-field was set to 0b0 and *GPD CommandID* = 0xE3, and if the proxy was a TempMaster, its dGP stub sends commands from its *gpTxQueue* to the GPD (for details, see sec. A.1.5.2.2); as described in step 9a, 9c – 9d. If *TransmitChannel* is equal to the operational channel; the proxy continues with step 6b.
   b. [215]If they are in commissioning mode, each proxy [216]forms a GP Commissioning Notification message, with *RxAfterTx* sub-field of the *Options* field set to 0b1; the sub-fields of the *Options* field set and the security fields set according to the security level of the triggering Channel Request GPDF, and the *GPD CommandID* and *GPD Command payload* copied from the received GPDF. Since the Channel Request GPDF in commissioning mode is always sent with *Frame type* field of the *NWK Frame Control* field set to 0b01 (Maintenance frame), the *GPD ID* field of the GP Commissioning Notification SHALL carry 0x00000000; the *ApplicationID* sub-field of the *Options* field SHALL be set to 0b000 and the *Endpoint* field is absent; any MAC

[211] Comment #5 from GP multi-sensor August PoC, Zigbee document 16-02611
[212] Comment #5 from GP multi-sensor August PoC, Zigbee document 16-02611
[213] Comment #3 from GP generic switch & compact attribute reporting SVE, May 2017
[214] CCB #2380; resolution added in 15-02014-010
[215] CCB #2378; Resolution added in 15-02014-011
[216] CCB #2378; Resolution added in 15-02014-011

source address information SHALL be ignored.
The Basic proxy, if the *Unicast communication* sub-field of the *Options* field of the GP Proxy Commissioning Mode was set to 0b0, sends the GP Commissioning Notification as broadcast on the operational channel, **with alias**, after *gppTunnelingDelay,* and with *BidirectionalCommunicationCapability* sub-field set to 0b0. If the *Unicast communication* sub-field of the *Options* field of the GP Proxy Commissioning Mode was set to 0b1, the Basic proxy sends the GP commissioning Notification as unicast to the originator of the GP Proxy Commissioning Mode command, on the operational channel, **without alias**, **i.e. with proxy's own address and sequence number**, after *Dmin_b,* and with *BidirectionalCommunicationCapability* sub-field set to 0b0.
The Advanced proxy, if the *Unicast communication* sub-field of the *Options* field of the GP Proxy Commissioning Mode was set to 0b0, sends the GP Commissioning Notification as broadcast on the operational channel **without alias, i.e. with proxy's own address and sequence number**, after *gppTunnelingDelay*, and the scheduled transmission SHOULD be dropped only if proxy receives the same frame within *gppTunnelingDelay* forwarded by a different proxy with *BidirectionalCommunicationCapability* sub-field set to 0b1, and better *GPP-GPD link* value (whereby better *GPP-GPD link* is defined as one having higher value of the *Link quality* sub-field, and if *Link quality* is equal, as one having higher value of the *RSSI* sub-field), or same *GPP-GPD link* value and lower short address. If the *Unicast communication* sub-field of the *Options* field of the GP Proxy Commissioning Mode was set to 0b1, the advanced proxy sends the GP Commissioning Notification as unicast to the originator of the GP Proxy Commissioning Mode command, on the operational channel, **without alias**, **i.e. with proxy's own address and sequence number**, after *gppTunnelingDelay,* and with *BidirectionalCommunicationCapability* sub-field set to 0b1.

7. **Sink receives GPD Channel Request command:** The sink receives a GPD Channel Request command (either directly or in a GP Commissioning Notification).
   a. If NOT in commissioning mode, the sink silently drops the command. **GOTO step 5,**
   b. If the sink received the GPDF in direct mode, and the *Frame Type* sub-field of the *NWK Frame Control* field was not set to 0b01, the sink SHALL drop the frame.
   GOTO step 5.
   c. [217]If the sink received the GPDF in direct mode and the *Frame Type* sub-field of the *NWK Frame Control* field was set to 0b01 and *GPD CommandID* = 0xE3, and if the sink was a TempMaster, its dGP stub sends commands from its gpTxQueue to the GPD (for details, see sec. A.1.5.2.2); as described in step 9. **GOTO step 7.d.**
   d. the sink appoints the TempMaster:
      i. If multi-hop commissioning and GP Basic sink: the sink can select the first proxy from which it receives the GP Commissioning Notification.
      If multi-hop commissioning and GP Advanced sink: the sink waits for *Dmax* to collect a couple of GP Commissioning Notification commands (from various proxies), selects the proxy with *BidirectionalCommunicationCapability* sub-field set to 0b1, if any, and from the remaining candidates one with to the best GPP-GPD link value (whereby better *GPP-GPD link* is defined as one having higher value of the *Link quality* sub-field, and if *Link quality* is equal, as one having higher value of the *RSSI* sub-field) and, if many, lowest address.
      ii. The sink generates the GPD Channel Configuration command, with the *OperationalChannel*

[217] CCB #2135; Resolution added in 15-02014-003; CCB #2743; Resolution added in 16-02607-0025.

sub-field of the *Channel* field carrying the operational channel of the network.
If EITHER the sink is a GP Basic sink OR the sink is a GP Advanced sink, but all of the candidate TempMasters are GP Basic proxies (as indicated by the *BidirectionalCommunicationCapability* sub-field of the *Options* field of the received GP Commissioning Notification set to 0b0), the sink SHALL set the *Basic* sub-field of the *Channel* field to 0b1.

iii. If the sink appoints itself as the TempMaster, it stores the Channel Configuration GPDF in its gpTxQueue, switches to (one of the) channel(s) the GPD will transmit the last Channel Request on in its next attempt(s), and enters receive mode.
It SHOULD broadcast GP Response command(s) with its own address in the *TempMaster short address* field.

iv. If one of the proxies is appointed as a TempMaster, the sink broadcasts (a) GP Response command(s) with the selected address of the TempMaster in the *TempMaster short address* field, the channel on which the TempMaster SHALL listen (always the last Channel Request during the next attempt) in the *TempMaster Tx channel* field, and with the GPD Channel Configuration command as payload. The *GPD ID* field of the GP Response carrying GPD Channel Configuration command SHALL carry the GPD ID 0x00000000, *ApplicationID* sub-field of the *Options* field SHALL be set to 0b000) and the *Endpoint* field is absent.
Note: *to improve the robustness of the procedure, the sink can appoint multiple TempMaster. It needs to make sure though, that their transmissions of Channel configuration GPDF will not collide, i.e. only one TempMaster per attempt, independent of the number of Channel Request transmissions in each attempt.*

v. If the sink is a GP Advanced sink and the TempMaster is GP Advanced as well, the sink may delay the transmission of the GP Response slightly.
This way, in the case where multiple sinks (possibly with different capabilities) are commissioned in parallel with the same GPD, the advanced sink can be the last one to nominate the TempMaster, and thus the GPD will continue with bidirectional commissioning procedure.

e. **GOTO step 8**.

8. **GP Response carrying GPD Channel Configuration command**: All proxies receive the GP Response (if sent) with the Channel Configuration GPDF:

a. The selected TempMaster sets its *FirstToForward* to TRUE, stores the Channel Configuration GPDF in its *gpTxQueue*, switches immediately to channel *TransmitChannel* with a 5s timeout, and enters receive mode.

b. [218]Other proxies remove any entries for GPD SrcID = 0x00000000 from their *gpTxQueue* (for details see sec. A.1.3.2.3), then silently drop the GP Response and remain on the operational channel. They set their *FirstToForward* to FALSE.

c. **GOTO step 3**.

9. **TempMaster transmits Channel Configuration GPDF:** The appointed TempMaster (proxy or sink) receives the Channel Request on channel *TransmitChannel*,

a. If the TempMaster receives any other GPDF than Channel Request GPDF on *TransmitChannel*, including a Commissioning GPDF or Success GPDF, it SHALL silently drop it.
If for the GPD Channel Request frame received on the *TransmitChannel*, the *Frame Type* sub-

[218] CCB #2380; Resolution added in 15-02014-010

field of the *NWK Frame Control* field NOT set to 0b01 (Maintenance frame) or the *Auto-commissioning* sub-field of the *NWK Frame Control* field is set to 0b1, the TempMaster SHALL silently drop the frame.

b. If proxy: SHALL NOT send a GP Commissioning Notification, neither on the operational channel nor on *TransmitChannel*;

c. TempMaster immediately switches to the Tx mode on channel *TransmitChannel*, and between *gpTxOffset* and *gpTxOffset+gpMaxTxOffsetVariation* ms after reception of the triggering GPDF (as measured on the medium) transmits at least one Channel Configuration GPDF
Note: the TempMaster can send the Channel Configuration GPDF several times (Channel Configuration GPFS), as long as the total GPFS duration does not exceed *gpTxDuration*.
The TempMaster SHALL send the Channel Configuration GPDF with *Frame Type* sub-field of the *NWK Frame Control* field set to 0b01 (Maintenance frame), unprotected and the *GPD ID* and *Endpoint* field absent; it SHALL send it in response to any Channel Request GPDF sent with *Frame Type* sub-field of the *NWK Frame Control* field set to 0b01 and *Auto-Commissioning* sub-field of the *NWK Frame Control* field set to 0b0; MAC source address information, if any, SHALL be ignored; the MAC Destination address field SHALL be set to 0xffff.

d. TempMaster returns to operational channel in commissioning mode.

e. If no GPD Channel Request command is received on channel *TransmitChannel* for 5sec, the TempMaster removes the Channel Configuration GPDF from its gpTxQueue and returns to the operational channel in commissioning mode. **GOTO step 4 (proxy) or step 5 (sink)**.

10. **GPD receives Channel Configuration GPDF:** The GPD receives the Channel Configuration GPDF, and if the frame is correctly formatted (*Frame Type* = 0b01, *Auto-Commissioning* = 0b0, *Extended NWK Frame Control* = 0b0, *SrcID*, *Endpoint*, *Security Frame counter* and *MIC* fields absent), the GPD stores the operational channel and sets its *ToggleChannel* internal variable to FALSE. The GPD MAY store the information whether the infrastructure supports the bidirectional communication in operation, as indicated by the *Basic* sub-field of the *Channel* field of the received Channel Configuration GPDF. **GOTO step 3**.
If the frame is incorrectly formatted, the GPD drops it without further processing.
11. All proxies and sinks receiving the Channel Configuration GPDF silently drop it. **GOTO step 3**.

Commissioning part

12. **Proxy receives commissioning command:** The proxies (also in combos) receiving a Commissioning GPDF, [219]Application Description GPDF, any other GPD command from the GPD CommandID range 0xE5 – 0xEF, any GPD command from the GPD CommandID range 0xB0 – 0xBF, or Data GPDF with *Auto-Commissioning* = 0b1 on the operational channel:

a. If for *ApplicationID* = 0b000 the SrcID was set to 0x00000000 or for *ApplicationID* = 0b010 the GPD IEEE address was set to 0x0000000000000000, the proxy SHALL silently drop the frame. **GOTO step 4.**
If *Auto-Commissioning* sub-field was set to 0b1 in a GPDF carrying GPD Commissioning command (i.e. with *GPD CommandID* 0xE0): silently drop the frame. **GOTO step 4.**
If *RxAfterTx* sub-field was set to 0b1 in a Data GPDF (see definition in sec. 3.4) with *Auto-Commissioning* sub-field set to 0b1: silently drop the frame. **GOTO step 4.**

b. If the GPDF was protected, all the proxy SHALL security-check and security-process it (see sec.

[219] Dec 2016 SVE comment: https://workspace.zigbee.org/kws/groups/PRO_GP/comments/view_comment?comment_id=1026

A.3.7.3, A.1.5.3).

i. If security processing fails on a proxy, the proxy SHALL forward the frame with *SecurityProcessingFailed* sub-field of the *Options* field of the GP Commissioning Notification set to 0b1.

ii. [220]In the current version of the specification, the proxy SHALL accept unprotected commissioning GPDF in commissioning mode, including subsequent commissioning, i.e. when the proxy already has a Proxy Table entry for this GPD with non-zero *SecurityLevel*. **GOTO step 12.c.**

iii. Otherwise, if security processing succeeds, the proxy proceeds with step c).

c. If *RxAfterTx* = 0b1 and *GPD CommandID* [221]set to 0xE0 [222]or 0xE4 or any other value from the range 0xE5 = 0xEF or [223]0xB0 – 0xBF, all proxies check if they have a GPDF in the gpTxQueue for this GPD (and *Endpoint*, specific or 0xff, if *ApplicationID* = 0b010); for details, see A.1.5.2.2.
If a proxy finds a frame for this GPD in its gpTxQueue (i.e. it is the TempMaster), its GP stub sends at least one Commissioning Reply GPDF between *gpTxOffset* and *gpTxOffset+gpMaxTxOffsetVariation* ms after reception of the triggering GPDF (as measured on the medium) on the operational channel, without CSMA/CA, using the same security level as the triggering GPDF. The transmission SHALL NOT take longer than *gpTxDuration*.
Note: (MAC ACK SHALL NOT *be requested).*

d. The proxy checks if it already has a Proxy Table entry for this GPD:

i. If yes, the settings of the *EntryActive*/*EntryValid* flags remain unchanged; the *InRange* flag is set to 0b1;
[224]When receiving an unprotected GPDF from a GPD for which the proxy already has an active valid Proxy Table entry with non-zero *SecurityLevel*, the proxy SHALL NOT update the *GPD security frame counter* field of this entry: NOT with a value of the MAC sequence number field of the triggering GPDF and NOT with the value of the *GPDoutgoingCounter* field if present in the payload of the unprotected Commissioning GPDF.
[225]When receiving a commissioning GPDF not carrying security frame counter (e.g. the Application Description GPDF), the proxy SHALL NOT store any value from that frame as the *GPD security frame counter* for this GPD.

ii. If not, the proxy creates an active invalid Proxy Table entry for this GPD, and updates it with all GPD capability information available from the GPDF, sets the *InRange* flag to 0b1, and sets the remaining capability fields to their default values.
A Basic Proxy is not required to create an active invalid Proxy Table entry.

e. All proxies form a GP Commissioning Notification message with *SecurityProcessingFailed* sub-field set to 0b0 and all available GPD capability information in the corresponding fields, to be sent on the operational channel. [226]Since the proxies are application-agnostic and the payload of the GPD commands is opaque to them, the payload of the GPD Commissioning command SHALL be included in its entirety and unmodified. I.a., even if the proxy stores the *gpLinkKey* attribute, the security key, if encrypted (as indicated by the *GPDkeyEncryption* sub-field of the

[220] Dec 2016 SVE comment: https://workspace.zigbee.org/kws/groups/PRO_GP/comments/view_comment?comment_id=1012
Dec 2016 SVE comment: https://workspace.zigbee.org/kws/groups/PRO_GP/comments/view_comment?comment_id=1027
[221] CCB #2447; as described in 17-02671-004; resolution added in 15-02014-012
[222] Dec 2016 SVE comment: https://workspace.zigbee.org/kws/groups/PRO_GP/comments/view_comment?comment_id=1026
[223] CCB #2447; as described in 17-02671-004; resolution added in 15-02014-012
[224] Dec 2016 SVE comment: https://workspace.zigbee.org/kws/groups/PRO_GP/comments/view_comment?comment_id=1024
[225] Dec 2016 SVE comment: https://workspace.zigbee.org/kws/groups/PRO_GP/comments/view_comment?comment_id=1024
[226] CCB #2118: Resolution added in 15-02014-002

Extended Options field of the GPD Commissioning command set to 0b1), will be sent unmodified, and the *GPDkeyMIC* field will be included unmodified.

i. If *RxAfterTx*=TRUE:
The Basic proxy, if the *Unicast communication* sub-field of the *Options* field of the GP Proxy Commissioning Mode was set to 0b0, SHALL send the GP Commissioning Notification as broadcast, **with** [227]**derived alias** (even if the proxy has a Proxy Table entry with assigned alias for this GPD), after *gppTunnelingDelay*, and with *BidirectionalCommunicationCapability* sub-field set to 0b0. If the *Unicast communication* sub-field of the *Options* field of the GP Proxy Commissioning Mode was set to 0b1, the Basic proxy sends the GP Commissioning Notification as unicast to the originator of the GP Proxy Commissioning Mode command, **without alias**, **i.e. with proxy's own address and sequence number**, after *Dmin*_b, and with *BidirectionalCommunicationCapability* sub-field set to 0b0.
The Advanced proxy, if the *Unicast communication* sub-field of the *Options* field of the GP Proxy Commissioning Mode was set to 0b0, schedules the transmission of the GP Commissioning Notification as broadcast **with proxy's own address and sequence number** after *gppTunnelingDelay*, and with *BidirectionalCommunicationCapability* sub-field set to 0b1, which is to be dropped only if the proxy sees the same frame within *gppTunnelingDelay* forwarded by a different proxy with *BidirectionalCommunicationCapability* sub-field set to 0b1, and the *GPP-GPD link* field from the received command has a better value than measured by the receiving proxy on receipt of this GPDF (whereby better *GPP-GPD link* is defined as one having higher value of the *Link quality* sub-field, and if *Link quality* is equal, as one having higher value of the *RSSI* sub-field), or if the *GPP-GPD link* value is equal, if the value in the *GPP address* field is lower than this proxy's NWK. If the *Unicast communication* sub-field of the *Options* field of the GP Proxy Commissioning Mode was set to 0b1, the advanced proxy sends the GP Commissioning Notification as unicast to the originator of the GP Proxy Commissioning Mode command, **without alias**, **i.e. with proxy's own address and sequence number**, after *gppTunnelingDelay*, and with *BidirectionalCommunicationCapability* sub-field set to 0b1.
The TempMaster from the Channel Request phase SHALL use the shortest *gppTunnelingDelay* (as if its *FirstToForward* flag was set to 0b1).
GOTO step 13.

ii. If *RxAfterTx*=FALSE,
the GP Commissioning Notification is sent as broadcast, **with** [228]**derived alias** (even if the proxy has a Proxy Table entry with assigned alias for this GPD), after *Dmin_u* (see sec. A.3.6.3.1), if the *Unicast communication* sub-field of the *Options* field of the GP Proxy Commissioning Mode was set to 0b0. If the *Unicast communication* sub-field of the *Options* field of the GP Proxy Commissioning Mode was set to 0b1, the GP Commissioning Notification is sent as unicast to the originator of the GP Proxy Commissioning Mode command, **without alias**, **i.e. with proxy's own address and sequence number**, after *Dmin_u*.
GOTO step 13.

13. **Sink receives commissioning command:** The pairing sink receives a Commissioning GPDF or Data GPDF with *Auto-Commissioning* 0b1 on the operational channel (in GP Commissioning

[227] Dec 2016 SVE comment: https://workspace.zigbee.org/kws/groups/PRO_GP/comments/view_comment?comment_id=1012
[228] Dec 2016 SVE comment: https://workspace.zigbee.org/kws/groups/PRO_GP/comments/view_comment?comment_id=1012

Notification command or directly).

a. If not in commissioning mode, the sink silently drops the Commissioning GPDF**.**
If *Auto-Commissioning* sub-field was set to 0b1 in a GPDF carrying GPD Commissioning command (i.e. with *GPD CommandID* 0xE0): silently drop the frame. **GOTO step 5.**
If *RxAfterTx* sub-field was set to 0b1 in a Data GPDF (i.e. with *GPD CommandID* other than 0xE0) with *Auto-Commissioning* sub-field set to 0b1: silently drop the frame. **GOTO step 5.**
If for *ApplicationID* = 0b000 the GPD SrcID was set to 0x00000000 or for *ApplicationID* = 0b010 the GPD IEEE address was set to 0x0000000000000000, the sink SHALL silently drop the frame. **GOTO step 5.**
[229]If in the received GPD Application Description command either of the fields *Total number of reports* or *Number of reports* is set to 0x00, silently drop the frame. **GOTO step 5.**

b. If the sink received the GPDF in direct mode, and the frame was protected, the sink SHALL security-check and security process the incoming packet (as described in sec. A.3.7.3, A.1.5.3).
   i. [230]In the current version of the specification, the sink SHALL accept unprotected Commissioning GPDF in commissioning mode, including subsequent commissioning, i.e. when the sink already has a Sink Table entry for this GPD with non-zero *SecurityLevel*.
   [231]
   [233]When receiving, directly or in a GP Commissioning Notification, a commissioning GPD command not carrying security frame counter (e.g. the GPD Application Description command), the sink SHALL NOT store any value from that frame as the *GPD security frame counter* for this GPD.
   GOTO step 13.d.

c. If security processing fails, and also in the case of GPDF received in tunneled mode with *SecurityProcessingFailed* sub-field of the *Options* field of the GP Commissioning Notification set to 0b1, the behavior is vendor- and application-specific.

d. [234]If (i) the sink received the GPDF in direct mode and (ii) if security processing succeeds or if the GPDF was unprotected, and if (iii) *RxAfterTx* = 0b1 and the *Frame Type* sub-field of the *NWK Frame Control* field was set to 0b00 and if (iv) either *GPD CommandID* = 0xE0 or *GDP CommandID* = 0xE4, and if (v) the sink was a TempMaster, then its dGP stub sends commands from its *gpTxQueue* to the GPD (for details, see sec. A.1.5.2.2); MAC acknowledgement SHALL NOT be requested .
If GDP CommandID = 0xE0 - **GOTO step 13.e.**
If GDP CommandID = 0xE4 - **GOTO step 13.f.**

e. The sink checks if the minimum security level supported by the GPD, as indicated by the *SecurityLevelCapabilities* sub-field and the *GPDkeyEncryption* sub-field of the *Extended Options* field of the received Commissioning GPDF. The *SecurityLevelcapabilities* sub-field of the received GPD Commissioning command SHALL be equal to or larger than the *Minimal GPD Security Level* sub-field of the *gpsSecurityLevel* (see sec. A.3.3.2.6). If the *Protection with gpLinkKey* sub-field of the *gpsSecurityLevel* is set to 0b1, then the *GPDkeyEncryption* sub-field of the *Extended Options* field of the received Commissioning GPDF SHALL be set as well.
According to the current version of the specification, the sink SHALL NOT accept GPDs supporting *gpdSecurityLevel* = 0b00 or GPDs not supporting TC-LK protection, unless explicitly

[229] Comment #8 from GP multi-sensor August PoC, Zigbee document 16-02611
[230] Dec 2016 SVE comment: https://workspace.zigbee.org/kws/groups/PRO_GP/comments/view_comment?comment_id=1012
[231] Comment #11 from GP generic switch & compact attribute reporting SVE, May 2017
[233] Dec 2016 SVE comment: https://workspace.zigbee.org/kws/groups/PRO_GP/comments/view_comment?comment_id=1024
[234] CCB #2135; Resolution added in 15-02014-004

configured to do so, using *gpsSecurityLevel*.
If there is no match or if the minimum security level supported by the GPD is equal to 0b01, the sink silently drops the frame; further behavior is vendor- and application-specific.

f. the sink checks if GPD application functionality matches (see sec. A.3.6.2.1). If there is no match, the sink drops the frame; further behavior is vendor- and application-specific.

g. If GPD application functionality matches, the sink SHALL check the contents of the security-related fields of the Commissioning GPDF payload (see sec. A.1.5.3). I.a., the sink SHALL check the following: if the *gpdSecurityLevel* has value other than 0b00 AND the sink does not have a key for this GPD yet AND EITHER *RxAfterTx* is NOT set and the *GPDkey* is not included in the Commissioning GPDF OR *RxAfterTx* is set and neither the *GPDkey* field is present nor the *GPSecurityKeyRequest* sub-field is set, then the sink shall silently drop the frame. **GOTO step 5.**

   i. If the check fails the behavior is vendor- and application-specific.

   ii. If the check succeeds, the sink stores the supplied GPD capability information, including the security-related parameters in a Sink Table entry for this GPD and *Endpoint*, specific or 0xff, if *ApplicationID* = 0b010, and continues with step (h).
   Note: If the commissioning command is a Data GPDF with *Auto-Commissioning* flag set to 0b1, the sink SHALL use the following default values: *MACsequenceNumberCapability* = 0b0; *RxOnCapability* = 0b0; *FixedLocation* = 0b0; if the GPDF was protected, the *SecurityLevel* and *SecurityKey* used, otherwise *SecurityLevel* = 0b00 and *KeyType* = 0b000.

h. If the sink already had a Sink Table entry for this GPD, (and *Endpoint*, specific or 0xff, if *ApplicationID* = 0b010), the sink can decide based on the application state and the content of its Sink Table to add, update or remove the Sink Table entry; the exact behavior is application- and vendor-specific.

i. If Data GPDF with *Auto-Commissioning* 0b1 OR Commissioning GPDF with *RxAfterTx*=FALSE and *GPD Application Description command follows* sub-field of the *Application Information* field is set to 0b0 OR the last Application Description GPDF (as can be derived from the fields *Total number of reports*, *Number of reports* and *Report identifier*) having the *RxAfterTx* sub-field set to FALSE [235]and the sink received all GPD Application Description commands (as can be derived from the fields *Total number of reports*) and at least one GPD Commissioning command from this GPD – **GOTO step 19**.
[236]If the sink receives the last Application Description GPDF (as can be derived from the fields *Total number of reports*, *Number of reports* and *Report identifier*) having the *RxAfterTx* sub-field set to FALSE and the sink did not receive all GPD Application Description commands from this GPD or did not receive a GPD Commissioning command from this GPD – **GOTO step 5**.
[237]To increase the robustness of the commissioning process, the sink SHALL be capable of receiving the Application Description GPDFs out of order and in duplicate.

j. Else if
[238]the sink receives an Application Description GPDF having the *RxAfterTx* sub-field set to TRUE and the sink did not receive all GPD Application Description commands from this GPD (as can be derived from the fields *Total number of reports*) or did not receive a GPD Commissioning command from this GPD – **GOTO step 5**.

[235] Comment #5 from GP multi-sensor August PoC, Zigbee document 16-02611
[236] Comment #5 from GP multi-sensor August PoC, Zigbee document 16-02611
[237] Comment #777 from GP multi-sensor v0.7 letter ballot
[238] Comment #5 from GP multi-sensor August PoC, Zigbee document 16-02611

[239]To increase the robustness of the commissioning process, the sink SHALL be capable of receiving the Application Description GPDFs out of order and in duplicate.
Else if the sink receives Commissioning GPDF with *RxAfterTx*=TRUE OR Application Description GPDF with *RxAfterTx*=TRUE [240]and the sink received all GPD Application Description commands (as can be derived from the fields *Total number of reports*) and at least one GPD Commissioning command from this GPD,

i. The sink prepares the Commissioning Reply GPDF, carrying the parameters requested by the GPD in the Commissioning GPDF.
[241]If both the *GPDkey* of key type 0b100 (OOB key) is included in the Commissioning GPDF AND the *GPSecurityKeyRequest* sub-field of the *Options* field is set to 0b1 AND if the *gpSharedSecurityKeyType* attribute has value other than 0x00, the sink SHALL include in the Commissioning Reply a shared key, of the type as specified by the *gpSharedSecurityKeyType* attribute,; if the *GPDkeyEncryption* sub-field of the triggering Commissioning GPDF was set to 0b1, the key SHALL be sent encrypted, the *GPDkeyMIC* field and the *Frame Counter* field SHALL be included.
If the *GPDkey* of key type and value as in *gpSharedSecurityKeyType* and *gpSharedSecurityKey* attribute is included in the Commissioning GPDF AND the *GPSecurityKeyRequest* sub-field of the *Options* field is set to 0b1 AND if the *gpSharedSecurityKeyType* attribute has value other than 0x00, the sink SHALL NOT include any key in the Commissioning Reply, the key type SHALL be set to the value of the *gpSharedSecurityKeyType* attribute; *GPDkeyEncryption* SHALL be set to 0b0, and the *GPDkeyMIC* field and the *Frame Counter* field SHALL NOT be included.
If no parameters are requested, but *RxAfterTx*=TRUE, Commissioning Reply GPDF SHALL still be created, with only the *Options* field present. [242]In that case, the sink SHALL set the *SecurityLevel* and *KeyType* sub-fields of the *Options* field of the Commissioning Reply GPDF to the corresponding values from the *Extended Options* field from the payload of the triggering Commissioning GPDF.

ii. The sink appoints the TempMaster:
- If multi-hop commissioning and GP Basic sink: the sink can select the first proxy from which it receives the GP Commissioning Notification.
  If multi-hop commissioning and GP Advanced sink: the sink waits for *Dmax* to collect a couple of GP Commissioning Notification commands (from various proxies), selects the selects TempMaster as described in sec. A.3.6.2.3;
- If the sink appoints itself as the TempMaster, it stores the Commissioning Reply GPDF in its gpTxQueue, and enters receive mode.
  It SHOULD broadcast GP Response command(s) with its own address in the *TempMaster short address* field.
- If one of the proxies is appointed as a TempMaster, the sink broadcasts (a) GP Response command(s) with the selected address of the TempMaster in the *TempMaster short address* field, and with the GPD Commissioning Reply command as payload.
- **GOTO step 14**.

[239] Comment #777 from GP multi-sensor v0.7 letter ballot
[240] Comment #5 from GP multi-sensor August PoC, Zigbee document 16-02611
[241] Dec 2016 SVE comment: https://workspace.zigbee.org/kws/groups/PRO_GP/comments/view_comment?comment_id=1012
[242] CCB #2719; Resolution added in 16-02607-025

14. **GP Response carrying GPD Commissioning Reply command** [243]**or any command in the range 0xF7-0xFF or 0xB0 – 0xBF**: The proxies receiving the GP Response command with the Commissioning Reply [244]or any command in the range 0xF7-0xFF or 0xB0 – 0xBF (if sent):
    a. All but the appointed TempMaster set the *FirstToForward* to 0b0, the TempMaster sets the *FirstToForward* to 0b1;
    b. The appointed TempMaster constructs the [245]GPDF (taking the supplied GPD [246]command) and stores it in its *gpTxQueue*.
    c. Non-TempMaster proxies check if they have any entry in the *gpTxQueue* for this GPD, and – if so – remove it; for details see sec. A.1.3.2.3.
    d. **GOTO step 3**.

15. **GPD receives Commissioning Reply GPDF:** A GPD receiving a Commissioning Reply GPDF:
    a. checks if the *ApplicationID* value, and the GPD SrcID/GPD IEEE address matches its own, and, if so,
    b. stores in NVM the supplied commissioning parameters (e.g. channel, PANId, key); the key, if sent encrypted, SHALL only be stored if the decryption succeeds with the *Frame Counter* value as provided in the frame.

    c. The GPD SHALL only reset the security frame counter if on reception of GPD Commissioning Reply, its security frame counter has value larger than 0x80000000 AND the supplied security key has a value or type different than the currently used security key.
    d. Sets the *ParametersStored* flag to TRUE **GOTO step 3**.
16. All proxies and sinks receiving a Commissioning Reply GPDF ignore it. **GOTO step 3**.

17. **Proxy receives Success GPDF:** The proxies (also in combos) receiving a [247]GPDF with (i) *GPD CommandID* = 0xE2 (GPD Success command) in case of *SecurityLevel* = 0b10 or (ii) any encrypted *GPD CommandID* in case of *SecurityLevel* = 0b11:
    a. If they are NOT in commissioning mode and the Success GPDF was received from a GPD the proxy has no Proxy Table entry for, or Success GPDF was incorrectly protected GPDF from a GPD the proxy has a Proxy Table entry for: silently drop the Success GPDF. See also sec. A.3.5.2.3.
    If *Auto-Commissioning* sub-field was set to 0b1, the proxy SHALL silently drop the frame. **GOTO step 4.** If for *ApplicationID* = 0b000 the SrcID was set to 0x00000000 or for *ApplicationID* = 0b010 the GPD IEEE address was set to 0x0000000000000000, the proxy SHALL silently drop the frame. **GOTO step 4.**
    b. [248]If they are in commissioning mode and the Success GPDF was protected, all the proxy SHALL security-check and security-process it (see sec. A.3.7.3, A.1.5.3).
        i. [249]If security processing fails on a proxy or the proxy cannot perform security processing due to lack of security parameters for this GPD (as indicated by GP-DATA.indication with the

[243] CCB #2447; as described in 17-02671-004; resolution added in 15-02014-012
[244] CCB #2447; as described in 17-02671-004; resolution added in 15-02014-012
[245] CCB #2447; as described in 17-02671-004; resolution added in 15-02014-012
[246] CCB #2447; as described in 17-02671-004; resolution added in 15-02014-012
[247] CCB #2362 and #2375; resolution added in 15-02014-011
[248] CCB #2362; resolution added in 15-02014-011
[249] CCB #2362; resolution added in 15-02014-011

Status COUNTER_FAILURE, AUTH_FAILURE or UNPROCESSED), the proxy SHALL forward the frame in a GP Commissioning Notification message with *SecurityProcessingFailed* sub-field set to 0b1 and the other sub-fields of the *Options* fields derived from the triggering GPDF (see sec. A.3.3.4.3), with the values of the field *GPD CommandID* and *MIC* copied from the triggering GPDF; and the *GPD Command payload*, if available, copied from the triggering GPDF.

ii. [250]Otherwise, if security processing succeeds, the proxy proceeds with step (c), forwarding the frame with *SecurityProcessingFailed* sub-field set to 0b0.

c. All proxies form a GP Commissioning Notification message, to be sent on the operational channel, containing the GPD Success command ID (0xE2) in the *GPD Command ID* field and 0xff in the *GPD Command payload* field.
Since GPD *RxAfterTx*=FALSE,
the GP Commissioning Notification is sent as broadcast, **with** [251]**derived alias** (even if the proxy has a Proxy Table entry with assigned alias for this GPD), after *Dmin_u* (see sec. A.3.6.3.1), if the *Unicast communication* sub-field of the *Options* field of the GP Proxy Commissioning Mode was set to 0b0. If the *Unicast communication* sub-field of the *Options* field of the GP Proxy Commissioning Mode was set to 0b1, the GP Commissioning Notification is sent as unicast to the originator of the GP Proxy Commissioning Mode command, on the operational channel, **without alias**, **i.e. with proxy's own address and sequence number**, after *Dmin_u*. The proxy sets the *BidirectionalCommunicationCapability* sub-field according to its capabilities.
GOTO step 18.

18. **The sink receives Success GPDF:** the sink receiving a GPD Success command:

a. If the sink is NOT in commissioning mode: silently drop the Success GPDF.
If *Auto-Commissioning* sub-field was set to 0b1, the sink SHALL silently drop the frame. **GO-TO step 5.**
If for *ApplicationID* = 0b000 the GPD SrcID was set to 0x00000000 or for *ApplicationID* = 0b010 the GPD IEEE address was set to 0x0000000000000000, the sink SHALL silently drop the frame. **GOTO step 5.**

b. [252]The Success GPDF SHALL be protected as agreed for the operational mode of this GPD, i.e. the key of the type and – in case of a sink-supplied key, also key value – as indicated by the sink in the GPD Commissioning Reply command (see step 13.j.i)).
The sink SHALL always security-check it; and in case of either direct reception or reception in a GP Commissioning Notification command with its *SecurityProcessingFailed* sub-field of the *Options* field set to 0b1, the sink SHALL first security-process it (see sec. A.3.7.3, A.1.5.3), whereby the sink SHALL only accept a reset security frame counter value from the GPD if the security frame counter of this GPD was larger than 0x80000000 AND a new security key value and/or new security key type was delivered to this GPD in the GPD Commissioning Reply command.

i. If security processing fails, the commissioning failed. The behavior is vendor- and application-specific.

ii. Otherwise, if security processing succeeds, the sink proceeds with **step 18c**.

[250] CCB #2362; resolution added in 15-02014-011
[251] Dec 2016 SVE comment: https://workspace.zigbee.org/kws/groups/PRO_GP/comments/view_comment?comment_id=1012
[252] CCB #2121, CCB #2120: Resolution added in 15-02014-002

c. [253]The sink SHALL remove from the gpTxQueue from all entries for this GPD. **GOTO step 19.**

Commissioning finalization

19. The sink finalizes commissioning: Pairing sink:
    a. Provides commissioning success indication to the user.
    b. If not done before: Creates a Sink Table entry for the GPD, storing all the available GPD information. [254]The key type and value SHALL be as agreed for the operational mode of this GPD, i.e. in case of bidirectional commissioning, as indicated by the sink in the GPD Commissioning Reply command (see step 13.j.i.) or in case of unidirectional commissioning, the OOB key supplied by the GPD (see step 3.d.)
    c. If the sink supports Translation Table functionality: if not done before and if the sink does not have generic GPD Command Translation Table entries for all of the GPD Data commands implemented by this GPD which are also supported by this sink, the sink creates Translation Table entries for all of the GPD Data commands supported implemented by this GPD which are also supported by this sink (see sec. A.3.6.2.2).
    d. If required, assigns an AssignedAlias for the GPD.
    e. SHALL send Device_annce for the alias (derived or assigned) for the GPD, with the exception of lightweight unicast communication mode.
    [255]When creating a pairing for a **GPD supporting generic switch functionality (GPD CommandID 0x69 and/or 0x6a),** the sink SHOULD only send Device_annce when creating the Sink Table entry for a particular GPD (i.e. upon successful commissioning of the first button of that GPD); it SHOULD NOT send the Device_annce upon successful subsequent commissioning of the same GPD (i.e. when the Sink Table entry already exists), irrespective of whether the subsequent commissioning procedure immediately follows the first commissioning exchange or the subsequent commissioning is independently triggered.
    f. Sends GP Pairing with *AddSink*=0b1, *RemoveGPD* = 0b0.
    By default, the GP Pairing command is sent in broadcast with destination endpoint set to 0xf2, with the value of the *CommunicationMode* sub-field in the *Options* field as requested by the sink and the remaining fields copied from its Sink Table entry. If *gpsCommunicationMode* is groupcast, the sink adds its Green Power EndPoint to the corresponding APS group.
    If the security level is > 0b00, the sink SHALL include the *GPD key* field in the GP Pairing command, irrespective of the key type. [256]The key type and value SHALL be as agreed for the operational mode of this GPD, i.e. in case of bidirectional commissioning, as indicated by the sink in the GPD Commissioning Reply command (see step 13.j.i.) or in case of unidirectional commissioning, the OOB key supplied by the GPD (see step 3.d.)
    [257]When creating a pairing for a **GPD supporting generic switch functionality (GPD CommandID 0x69 and/or 0x6a),** the sink SHOULD only send GP Pairing command when creating the Sink Table entry for a particular GPD (i.e. upon successful commissioning of the first button of that GPD); it SHOULD NOT send the GP Pairing command upon successful commissioning of subsequent buttons of the same GPD (i.e. when the Sink Table entry already exists), irrespective of whether the commissioning procedure for the subsequent button immediately follows commissioning of the first button or the commissioning is independently

[253] Generic switch commissioning guidelines, Zigbee document 16-02604-004
[254] CCB #2121; Resolution added in 15-02014-002
[255] Generic switch commissioning guidelines, Zigbee document 16-02604-004
[256] CCB #2121; Resolution added in 15-02014-002
[257] Generic switch commissioning guidelines, Zigbee document 16-02604-004

triggered.

g. If the sink does NOT support the *Sink Table-based groupcast forwarding* functionality, the sink SHALL ***only*** send a GP Pairing Configuration if the pairing was created for a pre-commissioned group. The GP Pairing Configuration SHALL have the *Action* sub-field of the *Actions* field set to 0b001, the *Send GP Pairing* sub-field set to 0b0, the *CommunicationMode* sub-field of the *Options* field set to 0b10, the *GroupList* field present and carrying the GroupID the pairing was created for and the corresponding alias (assigned or derived), and the *Number of paired endpoints* field SHALL be set to 0xfe.
If the just paired endpoint(s) of the sink are a member of multiple groups and the group to pair with was not explicitly selected, GP Pairing Configuration command(s) for all those GroupIDs SHALL be sent. If the GPD Commissioning command resulting in creation of this pairing contained *Application Information*, the sink MAY include it in the GP Pairing Configuration command, if it fits in the command payload without requiring use of fragmentation.
The sink SHALL NOT send GP Pairing Configuration command for full or lightweight unicast or derived groupcast pairing.
If the pre-commissioned group pairing was created for a GPD supporting GPD Compact Attribute Reporting command, as indicated by the reception of the GPD Application Description command, the sink SHALL, after transmitting the GP Pairing Configuration command with *Action* sub-field of the *Actions* field set to 0b001, also transmit the [258]GP Pairing Configuration command(s) with *Action* sub-field of the *Actions* field set to 0b101 and *Send GP Pairing* sub-field of the *Actions* field set to 0b0, carrying all the stored Application Description data (minimum requirement is *MultiSensorCommissioningBufferSize*), at the speed of approx. 1 message per second.
[259]In case of a pairing for a **GPD supporting generic switch functionality (GPD CommandID 0x69 and/or 0x6a),** the sink [260]SHALL send GP Pairing Configuration command upon each successful commissioning of a button, with the *Switch information* field present and carrying information related to that button.

h. If the sink supports *Sink Table-based groupcast forwarding* functionality, the sink SHALL send a GP Pairing Configuration if the pairing was created for a pre-commissioned group. The GP Pairing Configuration SHALL have the *Action* sub-field of the *Actions* field set to 0b001, the *Send GP Pairing* sub-field set to 0b0, the *CommunicationMode* sub-field of the *Options* field set to 0b10, the *GroupList* field present and carrying the GroupID the pairing was created for and the corresponding alias (assigned or derived), and the *Number of paired endpoints* field SHALL be set to 0xfe.
If the just paired endpoint(s) of the sink are a member of multiple groups and the group to pair with was not explicitly selected, GP Pairing Configuration command(s) for all those GroupIDs SHALL be sent. If the GPD Commissioning command resulting in creation of this pairing contained *Application Information*, the sink MAY include it in the GP Pairing Configuration command, if it fits in the command payload without requiring use of fragmentation.
If the pre-commissioned group pairing was created for a GPD supporting GPD Compact Attribute Reporting command, as indicated by the reception of the GPD Application Description command, the sink SHALL, after transmitting the GP Pairing Configuration command with *Action* sub-field of the *Actions* field set to 0b001, also transmit the [261]GP Pairing Configuration command(s) with *Action* sub-field of the *Actions* field set to 0b101 and *Send GP Pairing* sub-

[258] GP multi-sensor LB v0.9 comment #973: https://workspace.zigbee.org/kws/groups/PRO_GP/comments/view_comment?comment_id=973
[259] Generic switch commissioning guidelines, Zigbee document 16-02604-004
[260] GP multi-sensor LB v0.9 comment #973: https://workspace.zigbee.org/kws/groups/PRO_GP/comments/view_comment?comment_id=973
[261] GP multi-sensor LB v0.9 comment #973: https://workspace.zigbee.org/kws/groups/PRO_GP/comments/view_comment?comment_id=974

field of the *Actions* field set to 0b0, carrying all the stored Application Description data (minimum requirement is *MultiSensorCommissioningBufferSize*), at the speed of approx. 1 message per second.
[262]In case of a pairing for a **GPD supporting generic switch functionality (GPD CommandID 0x69 and/or 0x6a),** the sink [263]SHALL send GP Pairing Configuration command upon each successful commissioning of a button, with the *Switch information* field present and carrying information related to that button.

i. (if required) the user puts the sink into operational mode. The sinks exiting commissioning mode SHALL remove any commissioning-related entries from the gpTxQueue.

j. (if required) the sink sends GP Proxy Commissioning Mode (with *Action* sub-field of the *Options* field set to 0b0 = exit). **GOTO step 20**.

20. **Other sinks finalize commissioning**: The sinks receiving the GP Pairing Configuration command (if sent), act as described in A.3.5.2.5. **GOTO step 21.**

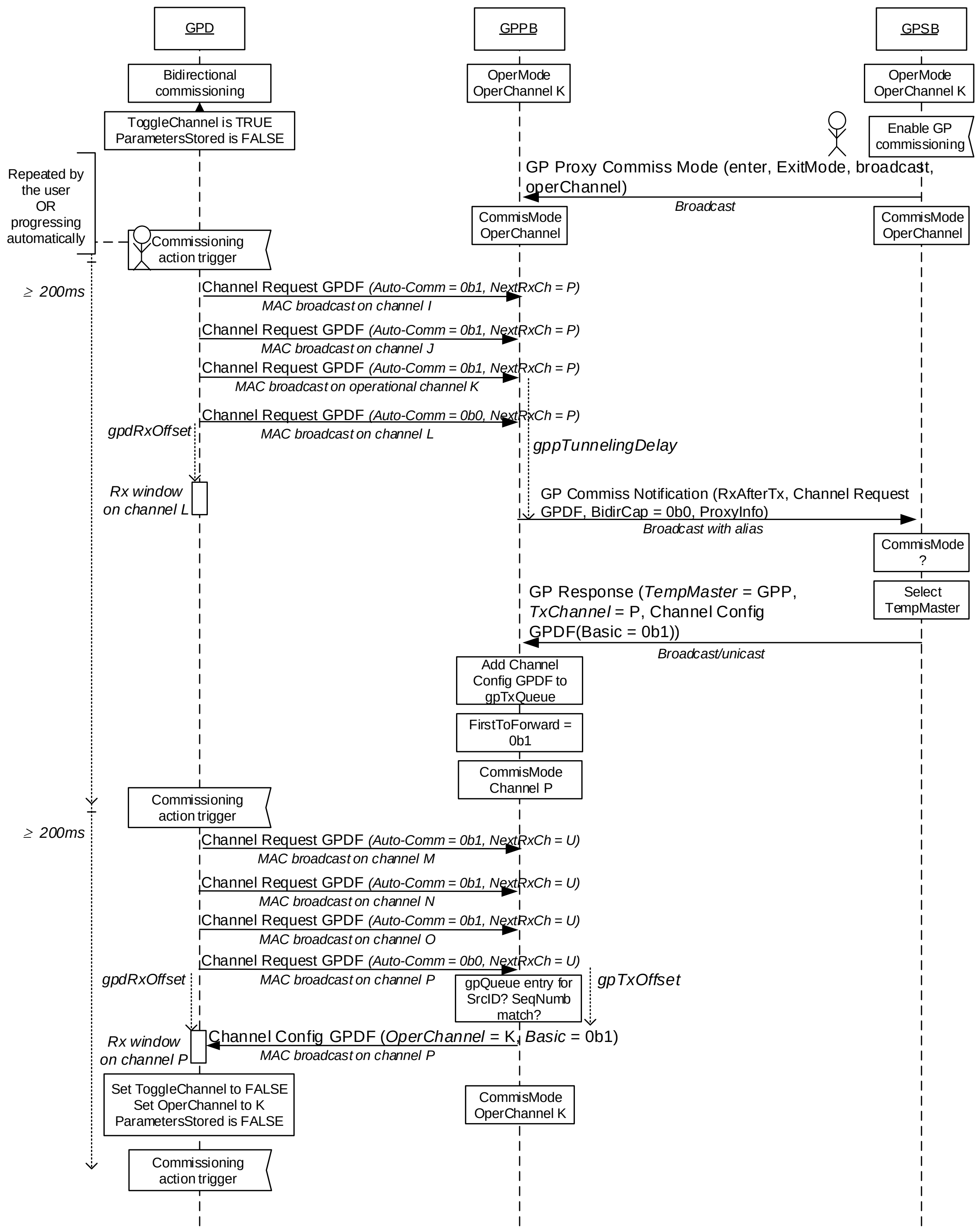
21. **Proxies finalize commissioning:** The proxies receiving the GP Pairing
    a. If the *SecurityLevel* sub-field of the *Options* field is set to 0b01, the proxy drops the GP Pairing, without creating Proxy Table entry;
    If for *ApplicationID* = 0b000 the GPD SrcID was set to 0x00000000 or for *ApplicationID* = 0b010 the GPD IEEE address was set to 0x0000000000000000, the proxy SHALL silently drop the frame; without creating Proxy Table entry.
    b. create/update Proxy Table entry;
    c. optionally, exit commissioning mode (if that was the *ExitMode* condition). The proxies exiting commissioning mode SHALL remove any commissioning-related entries from the gpTxQueue. **GOTO step 22**.

22. The proxies receiving GP Proxy Commissioning Mode with *Action* sub-field of the *Options* field set to 0b0 = exit (if sent) switch back to operational mode. The proxies exiting commissioning mode SHALL remove any commissioning-related entries from the gpTxQueue. **GOTO step 23**.
23. **GPD finalizes commissioning:** (if required) the user puts the GPD into operational mode. Then (or latest on first transmission of Data GPDF), the GPD sets its internal variables *ToggleChannel* to TRUE and *ParametersStored* to FALSE.
    [264]For a **GPD supporting generic switch functionality (GPD CommandID 0x69 and/or 0x6a),** the user may choose instead to progress directly to commissioning of a subsequent button. The internal variables *ToggleChannel* and *ParametersStored* are then set according to the commissioning method chosen (see step 3 above).
24. [265]**Sink finalizes commissioning:** when exiting commissioning mode, the sink SHALL remove any information on GPD for which the commissioning process didn't complete, incl. GPD for which only incomplete Application Description was received, even if the received part results in application functionality match. Further, the sink exiting commissioning mode SHALL remove any commissioning-related entries from its gpTxQueue.

[262] Generic switch commissioning guidelines, Zigbee document 16-02604-004
[263] GP multi-sensor LB v0.9 comment #973: https://workspace.zigbee.org/kws/groups/PRO_GP/comments/view_comment?comment_id=974
[264] Generic switch commissioning guidelines, Zigbee document 16-02604-004
[265] Comment #5 from GP multi-sensor August PoC, Zigbee document 16-02611

Figure 105 and Figure 106 depict an exemplary message sequence chart for multi-hop commissioning of a GPD capable of bidirectional commissioning (proxy and sink support bidirectional commissioning).

6747


GPD
GPPB
GPSB
Bidirectional commissioning
OperMode OperChannel K
OperMode OperChannel K
ToggleChannel is TRUE ParametersStored is FALSE
Enable GP commissioning
Repeated by the user OR progressing automatically
GP Proxy Commiss Mode (enter, ExitMode, broadcast, operChannel)
Broadcast
CommisMode OperChannel
CommisMode OperChannel
Commissioning action trigger
≥ 200ms
Channel Request GPDF (Auto-Comm = 0b1, NextRxCh = P)
MAC broadcast on channel I
Channel Request GPDF (Auto-Comm = 0b1, NextRxCh = P)
MAC broadcast on channel J
Channel Request GPDF (Auto-Comm = 0b1, NextRxCh = P)
MAC broadcast on operational channel K
Channel Request GPDF (Auto-Comm = 0b0, NextRxCh = P)
MAC broadcast on channel L
gpdRxOffset
gppTunnelingDelay
Rx window on channel L
GP Commiss Notification (RxAfterTx, Channel Request GPDF, BidirCap = 0b0, ProxyInfo)
Broadcast with alias
CommisMode ?
Select TempMaster
GP Response (TempMaster = GPP, TxChannel = P, Channel Config GPDF(Basic = 0b1))
Broadcast/unicast
Add Channel Config GPDF to gpTxQueue
FirstToForward = 0b1
CommisMode Channel P
Commissioning action trigger
≥ 200ms
Channel Request GPDF (Auto-Comm = 0b1, NextRxCh = U)
MAC broadcast on channel M
Channel Request GPDF (Auto-Comm = 0b1, NextRxCh = U)
MAC broadcast on channel N
Channel Request GPDF (Auto-Comm = 0b1, NextRxCh = U)
MAC broadcast on channel O
Channel Request GPDF (Auto-Comm = 0b0, NextRxCh = U)
MAC broadcast on channel P
gpdRxOffset
gpQueue entry for SrcID? SeqNumb match?
gpTxOffset
Rx window on channel P
Channel Config GPDF (OperChannel = K, Basic = 0b1)
MAC broadcast on channel P
Set ToggleChannel to FALSE Set OperChannel to K ParametersStored is FALSE
CommisMode OperChannel K
Commissioning action trigger

6748 [266]**Figure 105 – Exemplary MSC for multi-hop commissioning for bidirectional commissioning capable GPD, Basic Proxy and Basic Sink (part 1)**
6749

[266] CCB #2420; resolution added in 15-02014-010

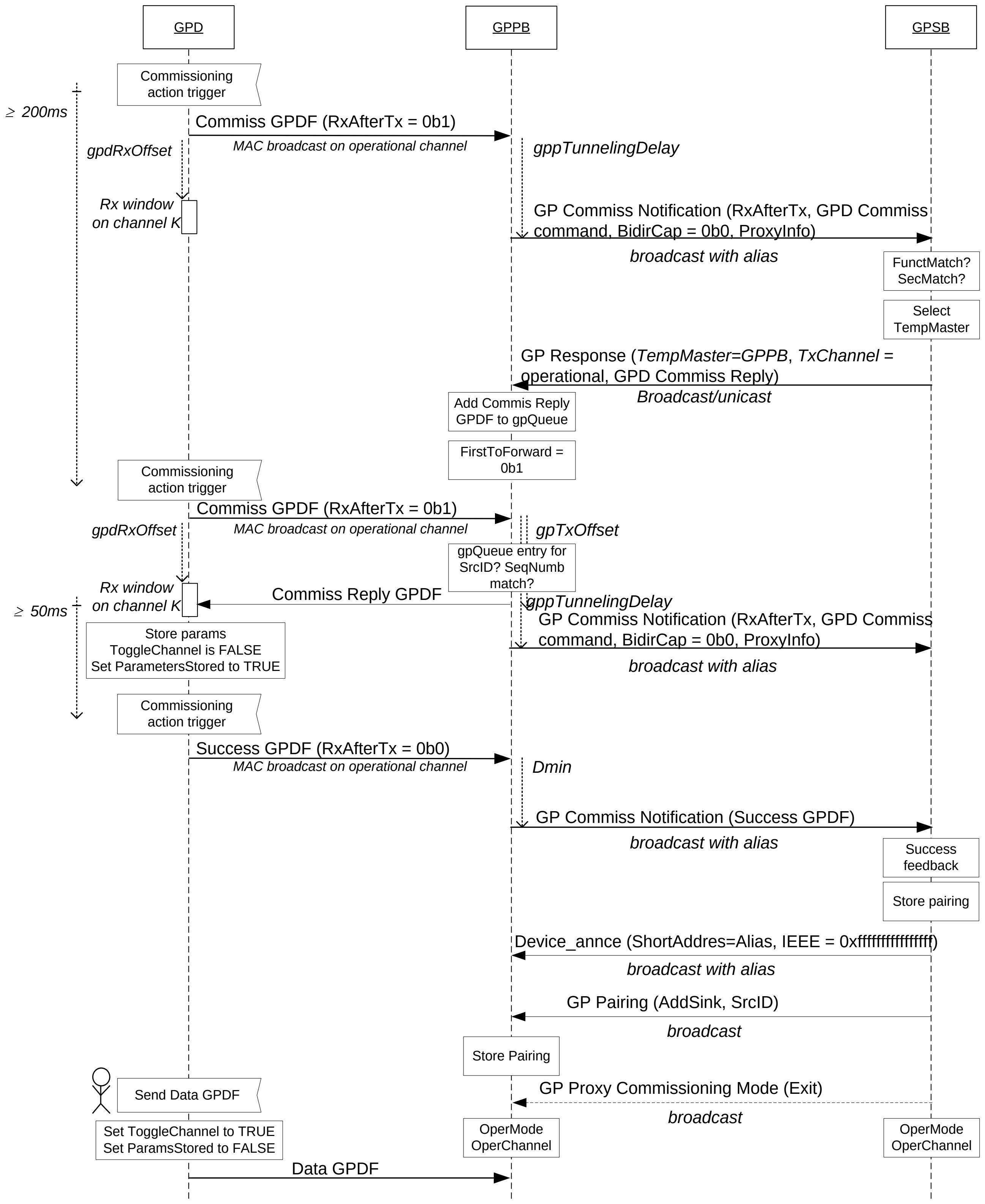

GPD
GPPB
GPSB
Commissioning action trigger
≥ 200ms
Commiss GPDF (RxAfterTx = 0b1)
MAC broadcast on operational channel
gpdRxOffset
gppTunnelingDelay
Rx window on channel K
GP Commiss Notification (RxAfterTx, GPD Commiss command, BidirCap = 0b0, ProxyInfo)
broadcast with alias
FunctMatch? SecMatch?
Select TempMaster
GP Response (TempMaster=GPPB, TxChannel = operational, GPD Commiss Reply)
Broadcast/unicast
Add Commis Reply GPDF to gpQueue
FirstToForward = 0b1
Commissioning action trigger
Commiss GPDF (RxAfterTx = 0b1)
MAC broadcast on operational channel
gpdRxOffset
gpTxOffset
gpQueue entry for SrcID? SeqNumb match?
Rx window on channel K
Commiss Reply GPDF
≥ 50ms
gppTunnelingDelay
GP Commiss Notification (RxAfterTx, GPD Commiss command, BidirCap = 0b0, ProxyInfo)
broadcast with alias
Store params ToggleChannel is FALSE Set ParametersStored to TRUE
Commissioning action trigger
Success GPDF (RxAfterTx = 0b0)
MAC broadcast on operational channel
Dmin
GP Commiss Notification (Success GPDF)
broadcast with alias
Success feedback
Store pairing
Device_annce (ShortAddres=Alias, IEEE = 0xffffffffffffffff)
broadcast with alias
GP Pairing (AddSink, SrcID)
broadcast
Store Pairing
Send Data GPDF
GP Proxy Commissioning Mode (Exit)
broadcast
Set ToggleChannel to TRUE Set ParamsStored to FALSE
OperMode OperChannel
OperMode OperChannel
Data GPDF

Figure 106 – Exemplary MSC for multi-hop commissioning for bidirectional commissioning capable GPD, Basic Proxy and Basic Sink (part 2)

A.3.9.2 Security commissioning best practices

A.3.9.2.1 GP infrastructure device commissioning

A.3.9.2.1.1 Proxy

When a proxy receives in commissioning mode:

- an unprotected Data GPDF with *Auto-Commissioning* sub-field set to 0b1 or unprotected Commissioning GPDF; the proxy schedules transmission of GP Commissioning Notification with the fields *GPD CommandID* and *GPD Command Payload* copied from the received GPDF, and the sub-fields of the *Options* fields set as follows: *SecurityLevel* 0b00, *SecurityKeyType* 0b000, *SecurityProcessingFailed* set to 0b0.
- a protected Data GPDF with *Auto-Commissioning* sub-field set to 0b1 or protected Commissioning GPDF:
  - and the proxy has the key and security processing succeeds (see A.3.7.3.1.1), the proxy schedules transmission of GP Commissioning Notification with the fields *GPD security key* and *GPD security frame counter* of the GP Commissioning Notification command payload present and carrying the values used for successful security processing and the sub-fields of the *Options* field are set as follows: *SecurityLevel* copied from the *Extended NWK Frame Control* field of the GPDF, *SecurityKeyType* of the key successfully used for security processing of the GPDF, *SecurityProcessingFailed* sub-field set to 0b0, [46] and *GPD key present* set to 0b1;
    the GPD CommandID and GPD Command Payload are then included in the clear.
    The Proxy Table entry SHALL be updated with the new *GPD security Frame Counter* value.
  - and the proxy has the key, but the security processing fails (see A.3.7.3.1.1), the proxy schedules transmission of GP Commissioning Notification with the sub-fields of the *Options* field are set as follows: *SecurityLevel* copied from the *Extended NWK Frame Control* field of the GPDF; *SecurityKeyType* set to 0b000 if the *SecurityKey* sub-field of the *Extended NWK Frame Control* field of the GPDF was set to 0b0 and 0b111 if the *SecurityKey* sub-field of the *Extended NWK Frame Control field* of the GPDF was set to 0b0; *SecurityProcessingFailed* set to 0b1, and *GPD key present* set to 0b0.
    the *GPD CommandID* and *GPD Command Payload* carrying unmodified values from the GPDF, *MIC* field present and carrying the value copied from the GPDF; *GPD security Frame Counter* carrying the value copied from the GPDF.
    The Proxy Table entry SHALL NOT be updated with the new *GPD security Frame Counter* value.
- the proxy does not have the key, it SHOULD drop the GPDF.

A.3.9.2.1.2 Sink

The following applies to GPD command used for commissioning, either received directly or tunneled in the GP Commissioning Notification with *SecurityProcessingFailed* sub-field of the *Options* field set to 0b0:

- If it was an unprotected Data GPDF with *Auto-Commissioning* bit set to 0b1, the check is successful if the *gpsSecurityLevel* attribute has the value of 0b00, and fails otherwise;
- if it was an unprotected Commissioning GPDF with none of the security related sub-fields of the *Options* or *Extended Options* fields (*GPsecurityKeyRequest, KeyType or GPDkeyPresent*) set, the check is successful if
  - both the *SecurityLevelCapabilities* sub-field of the *Extended Options* field, and *gpsSecurityLevel* attribute have the value of 0b00;
  - the check fails otherwise.

- If it was a protected Data GPDF with *Auto-Commissioning* bit set to 0b1 the check is successful if each of the following conditions is met:
  - the *SecurityLevel* of the *Extended NWK Frame Control* field is equal or higher to *gpsSecurityLevel* attribute, the key type as indicated by the *SecurityKey* sub-field is correct, and the key for this GPD is known to the sink. The check fails if at least one of the above conditions is not met.
- If it was a (protected or unprotected) Commissioning GPDF and the value of the *SecurityLevelCapabilities* sub-field in the *Extended Options* field is equal to or higher than *gpsSecurityLevel*, and:
  - the *KeyType* sub-field of the *Extended Options* field corresponds to NWK key or GP group key, and the *GPDoutgoingCounter* field is present, the check succeeds.
    If the *GPsecurityKeyRequest* (and *RxAfterTx*) was also set, the sink SHALL NOT include the key in GPDF Commissioning Reply frame. [267]The sink SHALL set the *SecurityLevel* and *KeyType* sub-fields of the *Options* field of the generated Commissioning Reply GPDF to the corresponding values from the *Extended Options* field from the payload of the triggering Commissioning GPDF.
  - the *KeyType* field of the *Extended Options* field corresponds to OOB individual key or Derived individual GPD key and the fields *GPDkey* and *GPDoutgoingCounter* are present, the check succeeds.
    If the *GPsecurityKeyRequest* (and *RxAfterTx*) was also set, the sink MAY include the key in GPDF Commissioning Reply frame. [268]The sink SHALL set the *SecurityLevel* and *KeyType* sub-fields of the *Options* field of the generated Commissioning Reply GPDF to the corresponding values from the *Extended Options* field from the payload of the triggering Commissioning GPDF.
  - If the *KeyType* sub-field of the *Extended Options* field has the value of 0b000, and the *GPsecurityKeyRequest* (and *RxAfterTx*) is also set, the check succeeds. The sink SHALL include the key in GPDF Commissioning Reply frame.
  - If the *GPsecurityKeyRequest* was set to 0b1, but *RxAfterTx* was set to 0b0, or if *GPsecurityKeyRequest* was set to 0b1, but *SecurityLevelCapabilities* was set to 0b00, the check fails.

The behavior on check failure as in the cases listed above and on reception of GP Commissioning Notification with *SecurityProcessingFailed* sub-field set to 0b1, is application-specific and out-of-scope of this document.

A.3.9.2.2 GPD commissioning[269]

The GPD that supports security (*SecurityLevelCapabilities* > 0b00) has the following security configuration options for commissioning mode:

- If the GPD supports *gpdSecurityLevel* other than 0b00 AND it does not share the key with the infrastructure, it SHALL enable key establishment with the infrastructure. To this end, the GPD SHALL include the key in the *GPDkey* field of the GPD Commissioning command, it MAY also request a key (if the GPD has the energy for receiving Commissioning Reply GPDF containing a key and storing it) by setting both *RxAfterTx* sub-field of the *Extended NWK Frame Control* and *GPSecurityKeyRequest* sub-field of the *Options* field of the GPD Commissioning command to 0b1. Note: Overwriting the individual key by the sink requires the GPD to first send and then receive a long GPDF with the 16B security key.

[267] CCB #2719; Resolution added in 16-02607-025
[268] CCB #2719; Resolution added in 16-02607-025
[269] Generic switch commissioning guidelines, Zigbee document 16-02604-004

- If the GPD is capable of sending the Success GPDF and if in the commissioning process the GPD and the pairing sink agree on key usage, the Success GPDF SHALL be sent protected with the key as indicated in the Commissioning Reply GPDF.
  If the agreed security level is *gpSecurityLevel*=0b11, the GPD SHALL protect the Success GPDF using *gpSecurityLevel*=0b11;
- If the GPD is capable of sending the Commissioning GPDF and:
  - the GPD has a shared key, i.e. the NWK key (*gpSecurityKeyType* = 0b001) or a GPD group key *gpSecurityKeyType* = 0b010 or 0b011), the Commissioning GPDF SHALL be sent unprotected, and in the Commissioning command payload, the *GPDkey* field SHALL be present and the *Security Frame Counter* field SHALL be present and carry the full 4B value; the sub-fields *GPDkeyPresent* and *GPDoutgoingCounterPresent* of the *Extended Options* field SHALL be set to 0b1,; the TC-LK protection SHALL be used.
  - the GPD has an individual GPD key (*gpSecurityKeyType* = 0b100 or 0b111), the Commissioning GPDF SHALL be sent unprotected, and in the Commissioning command payload, the *GPDkey* field SHALL be present and the *Security Frame Counter* field SHALL be present and carry the full 4B value; the sub-fields *GPDkeyPresent* and *GPDoutgoingCounterPresent* of the *Extended Options* field SHALL be set to 0b1, ; the TC-LK protection SHALL be used.

- DEPRECATED: Otherwise, is the GPD is only capable of sending Data GPDF with *Auto-Commissioning* sub-field set to 0b1 and:
  - the GPD has any key (e.g. as a result of pre-configuration), the Data GPDF SHALL be sent protected with this key, using the supported *gpdSecurityLevel*; the sub-fields of the *Extended NWK Frame Control* field of the Data GPDF SHALL be set accordingly, the fields *MAC sequence number, GPD security frame counter*, if present, and *MIC* set accordingly.
  - the GPD does not have any key, the Data GPDF SHALL be sent unprotected and the sub-fields *SecurityLevel* and *SecurityKey* of the *Extended NWK Frame Control* field of the Data GPDF, if present, SHALL be set accordingly.

Application profiles can adapt those commissioning recommendations to their needs.

A.3.9.3 Recommended GPD security key types

To allow for GPD mobility while minimizing the maintenance, the following types of keys are recommended for securing the GPD communication:

- for GPDs with *RxOnCapability*=0b0:
  - (individual) out-of-the-box key.
    Puts minimum requirements on GPD's Tx/Rx capabilities and allows for simple commissioning procedures. In case of mobility MAY lead to additional delay.
    Requires the manufacturer to provide the GPDs with the (individual) keys.
- For GPDs with *RxOnCapability*=0b1 and the capability of receiving the security key:
  - *GPD group key*
    The *NWK-key derived GPD group key* (*gpSecurityKeyType* 0b011) is the default option; the key is readily available to any GP infrastructure device being part of the Zigbee network, which limits key maintenance and simplifies GPD mobility. Note: in the event of NWK key update, updating the key on the GPDs is required as well.
    Non-derived *GPD group key* (*gpSecurityKeyType* 0b010) can be used as well; each GP device will have to be configured with it.

- For high-security applications - *GPD individual key* (*gpSecurityKeyType* 0b111).
- It is recommended, that the key sent in the Commissioning Reply GPDF is encrypted with the *gpLinkKey* (see sec. A.3.3.3.3).
  A *gpLinkKey* other than the default TC-LK can be used, if all involved devices will be supplied with this key prior to commissioning.

Using the Zigbee NWK key for securing the GP communication is NOT recommended.

For basic key types properties and usage recommendations – see sec. Table 53.

A.4 Green Power cluster extensions: ApplicationID 0b000 and 0b010

A.4.1 GPD CommandIDs

Table 54 and Table 55 define GPD Command IDs for the GPD commands without and with payload, respectively; together with corresponding Zigbee ZCL cluster, cluster-specific command and attribute (if required), for *ApplicationID* of 0b000 and 0b010. A dash (-) indicates that there is no default mapping to a Zigbee cluster; N/A indicates that there is no corresponding Zigbee functionality.

The handling of the GroupID parameter of the GPD Recall Scene and GPD Store Scene commands is defined in sec. A.4.2.7.

The command range 0xf0 – 0xff is reserved for commands sent to the GPD. They are defined in Table 56.

Future version of this specification MAY define additional GPD Commands.

Section A.4.3 specifies which GPD commands need to be implemented by a particular GPD type. Table 22 specifies which GPD commissioning commands need to be implemented by a sink.

6913 **Table 54 – Payloadless GPDF commands sent by GPD**

GPD command		Mapping to Zigbee		
CommandID	**Command Name**	**Corresponding ClusterID**	**CommandID**	**Command Payload**
0x00	Identify	Identify	Identify	0x003c
0x01 – 0x0F	Reserved			
0x10	Recall Scene 0	Scenes	Recall Scene	GroupID, SceneID = 0
0x11	Recall Scene 1	Scenes	Recall Scene	GroupID, SceneID = 1
0x12	Recall Scene 2	Scenes	Recall Scene	GroupID, SceneID = 2
0x13	Recall Scene 3	Scenes	Recall Scene	GroupID, SceneID = 3
0x14	Recall Scene 4	Scenes	Recall Scene	GroupID, SceneID = 4
0x15	Recall Scene 5	Scenes	Recall Scene	GroupID, SceneID = 5
0x16	Recall Scene 6	Scenes	Recall Scene	GroupID, SceneID = 6
0x17	Recall Scene 7	Scenes	Recall Scene	GroupID, SceneID = 7
0x18	Store Scene 0	Scenes	Store Scene	GroupID, SceneID = 0
0x19	Store Scene 1	Scenes	Store Scene	GroupID, SceneID = 1
0x1A	Store Scene 2	Scenes	Store Scene	GroupID, SceneID = 2
0x1B	Store Scene 3	Scenes	Store Scene	GroupID, SceneID = 3
0x1C	Store Scene 4	Scenes	Store Scene	GroupID, SceneID = 4
0x1D	Store Scene 5	Scenes	Store Scene	GroupID, SceneID = 5
0x1E	Store Scene 6	Scenes	Store Scene	GroupID, SceneID = 6
0x1F	Store Scene 7	Scenes	Store Scene	GroupID, SceneID = 7
0x20	Off	On/Off	Off	N/A
0x21	On	On/Off	On	N/A
0x22	Toggle	On/Off	Toggle	N/A
0x23	Release	-		
0x24 – 0x2F	Reserved			
0x30 – 0x33	Defined in Table 55			
0x34	Level Control/Stop	Level Control	Stop	N/A
0x35 – 0x38	Defined in Table 55			
0x39 – 0x3F	Reserved			
0x40	Move Hue Stop	Color Control	Move Hue	Stop
0x41 – 0x44	Defined in Table 55			
0x45	Move Saturation Stop	Color Control	Move Saturation	Stop
0x46 – 0x4B	Defined in Table 55			
0x4C – 0x4F	Reserved			
0x50	Lock Door	Door Lock	Lock Door	N/A
0x51	Unlock Door	Door Lock	Unlock Door	N/A
0x52 – 0x5F	Reserved			

6914

GPD command		Mapping to Zigbee		
CommandID	Command Name	Corresponding ClusterID	CommandID	Command Payload
0x60	Press 1 of 1	N/A		
0x61	Release 1 of 1	N/A		
0x62	Press 1 of 2	N/A		
0x63	Release 1 of 2	N/A		
0x64	Press 2 of 2	N/A		
0x65	Release 2 of 2	N/A		
0x66	Short Press 1 of 1	N/A		
0x67	Short Press 1 of 2	N/A		
0x68	Short Press 2 of 2	N/A		
0x69-0x6a	Defined in Table 55			
0x6b-0x6f	Reserved			
0x70-0x9f	Reserved			
0xA0-0xE0	Defined in Table 55			
0xE1	Decommissioning	N/A		
0xE2	Success	N/A		
0xE3	Defined in Table 55			
0xE4-0xEF	Defined in Table 55			

Table 55 defines CommandIDs for commands with non-zero payload, for *ApplicationID* of 0b000 and 0b010.

6918 **Table 55 – GPDF commands with payload sent by GPD**

GPD command		Mapping to Zigbee		
CommandID	**Command Name**	**ClusterID**	**Command Name**	**Command payload**
0x30	Move Up	Level Control	Move Up	
0x31	Move Down	Level Control	Move Down	
0x32	Step Up	Level Control	Step Up	
0x33	Step Down	Level Control	Step Down	
0x35	Move Up (with On/Off)	Level Control	Move Up (with On/Off)	
0x36	Move Down (with On/Off)	Level Control	Move Down (with On/Off)	
0x37	Step Up (with On/Off)	Level Control	Step Up (with On/Off)	
0x38	Step Down (with On/Off)	Level Control	Step Down (with On/Off)	
0x41	Move Hue Up	Color Control	Move Hue Up	
0x42	Move Hue Down	Color Control	Move Hue Down	
0x43	Step Hue Up	Color Control	Step Hue Up	
0x44	Step Hue Down	Color Control	Step Hue Down	
0x46	Move Saturation Up	Color Control	Move Saturation Up	
0x47	Move Saturation Down	Color Control	Move Saturation Down	
0x48	Step Saturation Up	Color Control	Step Saturation Up	
0x49	Step Saturation Down	Color Control	Step Saturation Down	

6919

GPD command		Mapping to Zigbee		
CommandID	Command Name	ClusterID	Command Name	Command payload
0x4A	Move Color	Color Control	Move Color	
0x4B	Step Color	Color Control	Step Color	
0x69	8-bit vector: press	See sec. A.4.2.2.1		
0x6a	8-bit vector: release	See sec. A.4.2.2.1		
0xA0	Attribute Reporting	Copied from the triggering GPD command	ZCL Report attributes command	Copied from the triggering GPD command
0xA1	Manufacturer-Specific Attribute Reporting	Copied from the triggering GPD command	ZCL Report attributes command	Copied from the triggering GPD command
0xA2	Multi-Cluster Reporting	Copied from the triggering GPD command	ZCL Report attributes command	Copied from the triggering GPD command
0xA3	Manufacturer-specific Multi-Cluster Reporting	Copied from the triggering GPD command	ZCL Report attributes command	Copied from the triggering GPD command
0xA4	Request Attributes	Copied from the triggering GPD command	ZCL Read attributes command	Copied from the triggering GPD command
0xA5	Read Attributes Response	Copied from the triggering GPD command	ZCL Read attributes response command	Copied from the triggering GPD command
0xA6	ZCL Tunneling	Copied from the triggering GPD command	Copied from the triggering GPD command	Copied from the triggering GPD command
[270]0xA7	Reserved			
[271]0xA8	Compact Attribute Reporting	Derived from the triggering GPD command, using the information sent during commissioning	ZCL Report attributes command	Derived from the triggering GPD command, using the information sent during commissioning
0xA9 – 0xAE	Reserved			
0xAF	[272][273]Any of the GPD sensor commands 0xA0 – 0xA3	Copied from the triggering GPD command	ZCL Report attributes command	Copied from the triggering GPD command
0xB0-0xBF	Manufacturer-defined GPD commands (payload is manufacturer-specific)			
0xC0-0xDF	Reserved			
0xE0	Commissioning	N/A		
0xE3	Channel Request	N/A		
0xE4	Application Description	N/A		
0xE5 – 0xEF	Reserved			

6920 Table 56 – GPDF commands sent to GPD

GPD command		Mapping to Zigbee		
Command ID	Command name	ClusterID	CommandID	Command Payload
0xF0	Commissioning Reply	N/A		
0xF1	Write Attributes	N/A		
0xF2	Read Attributes	N/A		
0xF3	Channel Configuration	N/A		
0xF4 – 0xF5	Reserved for other commands sent to the GPD			

[270] Comment #783 from GP multi-sensor v0.7 letter ballot
[271] Comment #783 from GP multi-sensor v0.7 letter ballot
[272] Note: 0xAF is not used as a true GPD CommandID, but as a way to make the Translation Tables more compact.
[273] Comment #1 from GP multi-sensor August PoC, Zigbee document 16-02611

0xF6	ZCL Tunneling	N/A
0xF7 – 0xFF	Reserved for other commands sent to the GPD	

A.4.2 Format of individual commands

The payload of any GPD Data command sent by the GPD SHALL NOT exceed:

- For a GPD with *ApplicationID* = 0b000: 59 octets;
- For a GPD with *ApplicationID* = 0b010: 54 octets.

This limitation is introduced to avoid that a proxy forwarding the GPD Data command in a GP Notification is forced to use fragmentation, or drop the command, if fragmentation is not supported.

The maximum payload length was calculated assuming unicast source routing, NWK layer protection, NO APS protection; 5B buffer was subtracted for future extensions to the GP Notification command.

A.4.2.1 Commissioning commands

In addition to the GPD commands with payload specified below, the following payloadless GPD commands also belong to the commissioning commands: GPD Success and GPD Decommissioning (see Table 48).

Note: some of the commissioning commands can also be used in operation, to manage the GPD, for example GPD Channel Configuration, GPD Commissioning Reply, GPD Decommissioning.

The payload of any GPD commissioning command sent by the GPD SHALL NOT exceed:

- For a GPD with *ApplicationID* = 0b000: 55 octets;
- For a GPD with *ApplicationID* = 0b010: 50 octets.

This limitation is introduced to avoid that a proxy forwarding the GPD commissioning command in a GP Commissioning Notification is forced to use fragmentation, or drop the command, if fragmentation is not supported.

The maximum payload length was calculated assuming unicast source routing, NWK layer protection, NO APS protection; 5B buffer was subtracted for future extensions to the GP Commissioning Notification command.

A.4.2.1.1 GPD Commissioning command

The payload of the GPD Commissioning command is formatted as shown in Figure 107 and Figure 108[274].

Octets	1	1	0/1	0/16	0/4	0/4
Data Type	8-bit enumeration	8-bit bitmap	8-bit bitmap	Security Key	Unsigned 32-bit integer	Unsigned 32-bit integer
Field name	GPD DeviceID	Options	Extended Options	GPDkey	GPDkeyMIC	GPDoutgoingCounter

Figure 107 – Format of the GPD Commissioning command payload (part 1)

0/1	0/2	0/2	0/1	0/Variable	0/Variable	0/Variable
8-bit bitmap	16-bit enumeration	16-bit enumeration	Unsigned 8-bit integer	Sequence of unsigned 8-bit integer	Sequence of unsigned 8-bit integer	Sequence of unsigned 8-bit integer
Application information	ManufacturerID	ModelID	Number of GPD commands	GPD CommandID list	Cluster List	Switch information

Figure 108 – Format of the GPD Commissioning command payload (part 2)

[274] PoC comment #17 (Zigbee document 16-02601)

Any additional fields applied after the end of the GPD Commissioning command SHALL be ignored by the devices according to the current version of the specification. The fields and sub-fields as defined in the current version of the specification SHALL be processed.

The *Auto-Commissioning* sub-field of the *NWK Frame Control* field for the GPDF carrying the GPD Commissioning command SHALL always be set to 0b0. The *GPD CommandID* field SHALL carry the value 0xE0, indicating the GPD Commissioning command, as defined in Table 55.

A.4.2.1.1.1 GPD DeviceID field

The GPD DeviceID field is always present and it carries one of the DeviceID, as defined in [13]. [275]

[276]Depending on the DeviceID used, additional rules regarding inclusion of the fields *Number of GPD commands*, *GPD CommandID list*, the *Cluster List* and the *Switch Information* may apply; see sec. A.4.2.1.1.7 - A.4.2.1.1.10.

A.4.2.1.1.2 Options field

The *Options* field of the GPD Commissioning command has the format as specified in Figure 109.

Bits: 0	1	2	3	4	5	6	7
MACsequenceNumberCapability	RxOnCapability	Application information present	Reserved	PANId request	GPsecurityKeyRequest	FixedLocation	ExtendedOptionsPresent

Figure 109 – Format of the Options field of the GPD Commissioning command

The *MACsequenceNumberCapability* sub-field is a Boolean flag. If the value of this sub-field is 0b1, then it indicates the GPD uses incremental MAC sequence number. If the value of this sub-field is 0b0, then it indicates that the GPD uses random MAC sequence number.

The *RxOnCapability* sub-field is a Boolean flag. If set to 0b1, it indicates that the GPD has receiving capabilities in operational mode. If set to 0b0, it indicates that the GPD does not enable its receiver in operational mode.

The *Application information present* sub-field is a Boolean flag. If set to 0b1, it indicates that the *Application information* field is present. If set to 0b0, it indicates that the *Application information* field is absent.

The *PANId request* sub-field is a Boolean flag. If the value of this sub-field is 0b1, then the GPD requests to receive the PAN ID value of the network. If the value of this sub-field is 0b0, then the GPD does not request to receive the PAN ID value. This sub field SHALL be set to 0b0 on transmission and ignored on reception, if the *RxAfterTx* sub field of the *NWK Frame Control* field of the GPDF carrying the GPD Commissioning command is set to 0b0.

The *GPsecurityKeyRequest* sub-field is a Boolean flag. If the value of this sub-field is set to 0b1, then the GPD requests to receive the GP Security Key. If the value of this sub-field is 0b0, then the GPD does not request to receive the GP Security Key. This sub field SHALL be set to 0b0 on transmission and ignored on reception, if the *RxAfterTx* sub field of the *NWK Frame Control* field of the GPDF carrying the GPD Commissioning command is set to 0b0.

[275] Comment #773 from GP multi-sensor v0.7 letter ballot
[276] Comment #773 from GP multi-sensor v0.7 letter ballot

The *FixedLocation* sub-field is a Boolean flag. If the value of this sub-field is 0b0, then it indicates that the GPD can change its position during its operation in the network. If the value of this sub-field is 0b1, then the GPD is not expected to change its position during its operation in the network.

The *ExtendedOptionsPresent* sub-field is a Boolean flag. If the value of this sub-field is 0b1, then it indicates that the *Extended Options* field is present.

A.4.2.1.1.3 Extended Options field

The *Extended Options* field SHALL be present, if the GPD is capable of supporting security and it transmits and/or requests security settings.

The *Extended Options* field of the GPD Commissioning command has the format as specified in Figure 110.

Bits: 0-1	2-4	5	6	7
SecurityLevelCapabilities	KeyType	GPDkeyPresent	GPDkeyEncryption	GPDoutgoingCounterPresent

Figure 110 – Format of the *Extended Options* field of the GPD Commissioning command

The *SecurityLevelCapabilities* sub-field indicates the device's security capabilities during normal operation. It can take values as defined in Table 11.

According to the current version of the specification, only GPD that support *gpdSecurityLevel* = 0b10 or higher AND support TC-LK protection (as indicated by the *GPDkeyEncryption* sub-field of the *Extended Options* field of the GPD Commissioning command) of the GPD key, if exchanged over the air, can be certified.

When the *Extended Options* field is not present in the GPD Commissioning command and the *GPsecurityKeyRequest* sub-field of the *Options* field is set to 0b1, the 0b01 is taken as the default value. When the *Extended Options* field is not present in the GPD Commissioning command and the *GPsecurityKeyRequest* sub-field of the *Options* field is set to 0b0, the 0b00 is taken as the default value.

If *SecurityLevelCapabilities* sub-field is set to 0b00, then the *KeyType* sub-field SHALL be set to 0b000 on transmission and SHALL be ignored on reception. Furthermore, if *SecurityLevelCapabilities* sub-field is set to 0b00, then the *GPDkeyPresent* and *GPDoutgoingCounterPresent* SHALL be set to 0b0 on transmission and ignored upon reception, and the fields *GPDkey* and *GPDoutgoingCounter* field SHALL NOT be present on transmission and SHALL be ignored upon reception.

The *KeyType* sub-field indicates the type of the security key this GPD is configured with. The *KeyType* can take the values as defined in A.3.7.1.2.

When *GPDkeyPresent* sub-field is set to 0b1 and the *GPDKeyEncryption* sub-field is set to 0b0, the *GPDkey* field is present in the clear, and carries the *gpdSecurityKey*, of the type as indicated in the *gpdSecurityKeyType* parameter; the *GPDkeyMIC* field is absent. When *GPDkeyPresent* sub-field is set to 0b1 and the *GPDkeyEncryption* sub-field is set to 0b1, both fields *GPDkey* and *GPDkeyMIC* are present; the field *GPDkey* contains the *gpdSecurityKey*, of the type as indicated in the *gpdSecurityKeyType*, encrypted with the default TC-LK (see A.3.3.3.3) as described inA.3.7.1.2.3; and the *GPDkeyMIC* field contains the MIC for the encrypted GPD key, calculated as described in A.3.7.1.2.3. When *GPDkeyPresent* sub-field is set to 0b0, the *GPDKeyEncryption* sub-field indicates the GPD's capability of protecting the *GPDkey* field as described in A.3.7.1.2.3; if set to 0b1, the GPD is capable; if set to 0b0, it is not.

If the *GPDkeyPresent* sub-field is set to 0b1, the *GPDoutgoingCounterPresent* sub-field SHALL be set to 0b1 and the *GPDoutgoingCounter* field SHALL be present.

The *GPDoutgoingCounterPresent* sub-field, if set to 0b1, indicates that the *GPDoutgoingCounter* is present. If *GPDoutgoingCounter* field is present in the payload of the GPD Commissioning command (and it SHALL if *SecurityLevelCapabilities* sub-field of the *Extended Options* field is set to 0b10 or 0b11), the value it carries SHALL be incremented for every transmission of a Commissioning GPFS.

A.4.2.1.1.4 Application information field

The *Application information* field SHALL be present, if any of the Application Information fields: *ManufacturerID*, *ModelID*, *GPD CommandID list* and *Cluster list* are present.

Detailed rules for inclusion of those Application Information fields are defined in sections A.4.2.1.1.5 - A.4.2.1.1.9.

The *Application information* field of the GPD Commissioning command has the format as specified in Figure 111.

Bits: 0	1	2	3	4	5	6..7
ManufacturerID present	ModelID present	GPD commands present	Cluster list present	[277]Switch information present	GPD Application Description command follows	Reserved

Figure 111 – Format of the *Application information* field of the GPD Commissioning command

The *ManufacturerID present* sub-field is a Boolean flag. If set to 0b1, it indicates that the *ManufacturerID* field is present. If set to 0b0, it indicates that the *ManufacturerID* field is absent.

The *ModelID present* sub-field is a Boolean flag. If set to 0b1, it indicates that the *ModelID* field is present. If set to 0b0, it indicates that the *ModelID* field is absent.

The *GPD commands present* sub-field is a Boolean flag. If set to 0b1, it indicates that the fields *Number of GPD commands* and *GPD CommandID list* are present. If set to 0b0, it indicates that both those field are absent.

The *Cluster list present* sub-field is a Boolean flag. If set to 0b1, it indicates that the *Cluster List* field is present. If set to 0b0, it indicates that this field is absent.

[278]The *Switch information present* sub-field is a Boolean flag. If set to 0b1, it indicates that the *Switch information* field is present. If set to 0b0, it indicates that this field is absent.

The *GPD Application Description command follows* sub-field is a Boolean flag. If set to 0b1, it indicates that after the current Commissioning GPDF, the GPD Application Description command (0xE4, see sec. A.4.2.1.6) will follow. If set to 0b0, it indicates that the GPD Application Description command will not be sent after the current GPD Commissioning command.
The *GPD Application Description command follows* sub-field SHALL be set to 0b1 if the GPD supports the GPD Compact Attribute Reporting command ([279]0xA8, see sec. A.4.2.3.6).

A.4.2.1.1.5 ManufacturerID field

The *ManufacturerID* field can take values as defined in [7].

The *ManufacturerID* field SHALL be present, if the *ModelID* field is present, if the *GPD CommandID list* contains any manufacturer-specific GPD commands, or if the *Cluster List* field contains any manufacturer-specific clusters. In other cases, the *ManufacturerID* field MAY be present; the *ManufacturerID present* sub-field of the *Application information* field SHALL be set accordingly.

[277] PoC comment #25 (Zigbee document 16-02601)
[278] PoC comment #25 (Zigbee document 16-02601)
[279] Comment #783 from GP multi-sensor v0.7 letter ballot

A.4.2.1.1.6 ModelID field

The *ModelID* field carries a manufacturer-defined identification of the product type. If *ModelID* is present, the *ManufacturerID* SHALL be present as well; the sub-fields of the *Application information* field SHALL be set accordingly.

The *ModelID* field MAY be preset even if the *GPD CommandID list* and the *Cluster list* fields are absent and/or if the *DeviceID* carries a value other than 0xFE.

A.4.2.1.1.7 Number of GP commands field

The *Number of GP commands* defines the number of items in the *GP command list* field. This field SHALL have value always greater than zero otherwise the field SHALL NOT be present; the *GPD commands present* sub-field of the *Application information* field SHALL be set accordingly.

A.4.2.1.1.8 GPD CommandID list field

The *GPD CommandID list* contains the GPD commands used by this GPD.

The term **standard GPD Data commands** is used to refer to any GPD Data commands defined by the GP specification, transmitted (with CommandID from the range 0x00 – 0x9f, as listed in Table 54, Table 55 and Table 56) or received (with CommandID 0xF1, 0xF2, 0xF6, as listed in Table 56).

The term **standard GPD reporting commands** is used to refer to any GPD commands 0xA0 – 0xA3 and 0xA6, defined by the GP specification.

The *GPD CommandID list* SHALL be present:

- if a GPD with *DeviceID* = 0xFE implements any standard GPD Data commands, unless:
  - the GPD Compact Attribute Reporting is the only GPD Data command supported by the GPD;
  - [280]the *Cluster list* is present and not empty;
- if a GPD with DeviceID != 0xFE implements other standard GPD Data commands than mandated for its *DeviceID* (see [13]); i.e. adds or removes standard GPD Data commands.

The *GPD CommandID list* MAY be present in other cases.

If present, the *GPD CommandID list* SHALL contain all the standard GPD Data commands supported by that GPD transmitted and received; it SHALL NOT contain the GPD commissioning commands (see sec. A.4.2.1); the order of commands in the list is unspecified.

The *GPD CommandID list* MAY contain any manufacturer-defined GPD commands (i.e. CommandIDs from the range 0xB0 – 0xBF, see Table 55), also in addition to any standard GPD Data commands. If the *GPD CommandID list* contains any manufacturer-defined GPD commands, the *ManufacturerID* field SHALL be present.

The *GPD CommandID list* SHALL be consistent with the device PICS: only the functionality disclosed can be certified.

A number of examples below aims at clarifying the rules for *GPD CommandID list* field usage:

- If a GPD with *DeviceID* != 0xFE only implements GPD Data commands mandated for its *DeviceID*, the GPD is not required (but can) include the GPD CommandID list.

[280] CCB #2736; Resolution added in 16-02607-025;

- [281]If a GPD supporting ZCL clusters, as indicated by sensor *DeviceID* 0x30 – 0x33, implements only the standard GPD reporting commands, the GPD is not required (but can) include the GPD CommandID list.
- s
- If a GPD supporting ZCL clusters (as indicated by sensor *DeviceID* 0x30 – 0x33 or by including *Cluster list* field), implements any standard GPD Data commands in addition to the standard GPD reporting commands, the GPD is required to include all of those standard GPD Data commands in the *GPD CommandID list* field; it can also include the standard GPD reporting commands.

A.4.2.1.1.9 Cluster List field

The *Cluster List* field contains a list of server and client clusters supported by this particular GPD. The *Cluster List* field is formatted as specified in Figure 112.

Octets	1	Variable	Variable
Data Type	Unsigned 8-bit integer	Sequence of unsigned 16-bit integer	Sequence of unsigned 16-bit integer
Field name	Length of ClusterID list	Cluster ID List Server	ClusterID List Client

Figure 112 – Format of the Cluster List field

The *Length of ClusterID list* field specifies the number of 16-bit ClusterIDs server and client clusters in the *ClusterID list server/ ClusterID list client* field, respectively. The *Length of ClusterID list* field SHALL be formatted as shown in Figure 113. This field SHALL have value always greater than zero otherwise the *Cluster List* field SHALL NOT be present.

Bits: 0-3	4..7
Number of server ClusterIDs	Number of client ClusterIDs

Figure 113 – Format of the Length of ClusterID list field

The *ClusterID list server/client* field contains a list of ClusterIDs that are supported by this GPD in server and client role, respectively; the order of clusters in each list is unspecified.

The term **standard ZCL cluster** is used to refer to any cluster defined in the Zigbee Cluster Library [3], any standard commands and/or attributes of that cluster. Manufacturer-specific clusters are clusters using ClusterIDs from the manufacturer-specific range as defined in the ZCL [3].

[282]The *Cluster list* SHALL NOT include the functionality accessible exclusively via the GPD Compact Attribute Reporting command ([283]0xA8). If the GPD only supports cluster functionality accessible via the GPD Compact Attribute Reporting command, the *Cluster list* SHALL be omitted.
[284]A GPD MAY implement some functionality accessible via the GPD Compact Attribute Reporting command, in addition to some functionality accessible via other GPD commands. The GPD SHALL represent it correctly in the Commissioning GPDF and Application Description GPDF, and the sink SHALL process both parts.

[281] CCB #2736; Resolution added in 16-02607-025;
[282] Comment #4, #6, #13 from GP multi-sensor August PoC, Zigbee document 16-02611
[283] Comment #783 from GP multi-sensor v0.7 letter ballot
[284] Comment #1 from GP generic switch & compact attribute reporting SVE, May 2017

The *Cluster list* SHALL NOT include any functionality accessible exclusively via the GPD commands from the 0x00 – 0x9F and 0xB0 – 0xBF range. If the GPD only supports application functionality accessible via those commands, the *Cluster list* SHALL be omitted.

[285]The *Cluster list* SHALL only include the cluster functionality accessible using the following GPD commands: 0xA0 – 0xA6 and 0xF1, 0xF2, 0xF6. In addition, the following applies:

- The *Cluster list* SHALL be present if a GPD with *DeviceID* != 0xFE implements other standard ZCL clusters than mandated for its *DeviceID* (see [13]); i.e. adds standard ZCL clusters;
- The *Cluster list* MAY be included by GPD with *DeviceID* != 0xFE in other cases, e.g. it MAY list the clusters corresponding to its DeviceID;
- The *Cluster list* SHALL be present if a GPD with *DeviceID* = 0xFE supports any standard ZCL clusters; the *Cluster list* SHALL contain all the standard ZCL cluster supported by that GPD.
- If included, the *Cluster list* of a GPD with *DeviceID* != 0xFE SHALL contain all the additional standard ZCL clusters supported by that GPD; it MAY (but is not required to) contain other standard ZCL clusters than mandated for this *DeviceID*;
- The *Cluster list* MAY contain any manufacturer-specific clusters, also in addition to standard ZCL clusters. If the *Cluster list* contains any manufacturer-specific clusters, the *ManufacturerID* field SHALL be present.

The order of clusters in the *Server/Client list* is unspecified.

The *Cluster list* SHALL be consistent with the device PICS: only the functionality disclosed can be certified.

A.4.2.1.1.10 Switch information field

The *Switch information* field is formatted as specified in Figure 114.

Octets	0/1	0/1	0/1
Data Type	Unsigned 8-bit integer	8-bit bitmap	8-bit bitmap
Field name	Switch info length	Generic switch configuration	Current contact status

Figure 114 – Format of the *Switch information* field of the GPD Commissioning command payload

[286]The *Switch information* field SHALL only be present if the *Switch information present* sub-field of the *Application information* field is set to TRUE. That SHALL only be the case if:

- the DeviceID is set to 0x07;
- and/or CommandID 0x69/0x6a is included in the GPD command list of the *ApplicationInformation* block.

Otherwise, the *Switch information present* sub-field of the *Application information* field is set to FALSE and the *Switch information* field SHALL be absent.

The *Switch info length* field indicates the total length of the following switch configuration information, i.e. it carries the value 0x02 according to the current specification.

The *Generic switch configuration* field is formatted as shown in Figure 115.

[285] Comment #4, #6 from GP multi-sensor August PoC, Zigbee document 16-02611
[286] Comment #15 from GP multi-sensor August PoC, Zigbee document 16-02611

Bits: 0-3	4..5	6..7
Number of contacts	Switch type	Reserved

Figure 115 – Format of the Generic switch configuration field

The *Number of contacts* sub-field indicates the number of contacts supported by the module, between 0 and 8.

The *Switch type* sub-field indicates the type of physical switch actuation, and can take any of the non-reserved values from Table 57.

Table 57 – Values of the Switch type sub-field of the Generic switch configuration field

Value	Meaning
0b00	Unknown: exact configuration apart from number of contacts unknown
0b01	Button switch
0b10	Rocker switch
0b11	Reserved

The *Current contact status* field is formatted exactly like the *Contact status* field (see sec. A.4.2.2.1) and carries the current contact status information corresponding to the user action that triggered the sending of this particular Commissioning GPDF.

[287]Note: The GPD Commissioning command SHOULD NOT be sent with *Current contact status* field set to 0x00 and/or with the *Number of contacts* sub-field of the *Generic switch configuration* field set to 0x0, as from this information no meaningful Translation Table entries can be derived.

A.4.2.1.1.11 When generated

This frame is generated by the GPD to manage its status in the network, i.e. it MAY be used to manage, i.e. create, remove or update pairings.

A.4.2.1.1.12 Effect on receipt

On reception of GPD Commissioning command, a proxy acts as described in A.3.5.2.1or A.3.5.2.3, and a sink acts as described in A.3.5.2.5 or A.3.5.2.4.

A.4.2.1.2 Commissioning Reply command

The payload of the Commissioning Reply command is formatted as shown in Figure 116.

Octets	1	0/2	0/16	0/4	0/4
Data Type	8-bit bitmap	Unsigned 16-bit integer	Security key	Unsigned 32-bit integer	Unsigned 32-bit integer
Field name	Options	PANId	GPDsecurityKey	GPDkeyMIC	Frame Counter

Figure 116 – Format of the GPD Commissioning Reply command payload

If GPD uses *ApplicationID* 0b000, the *GPD SrcID* field of the Commissioning Reply frame SHALL carry the value of the GPD SrcID; if GPD uses *ApplicationID* 0b010, the MAC Destination address field SHALL carry the GPD IEEE address of the GPD to which this frame is being sent.

The *GPD CommandID* SHALL carry the value 0xF0, indicating the GP Commissioning Reply command, as defined in Table 56.

[287] PoC comment #4 (Zigbee document 16-02601)

A.4.2.1.2.1 Options field

The *Options* field is formatted as shown in Figure 117.

Bits: 0	1	2	3-4	5-7
PANID present	GPDsecuri-tyKeyPresent	GPDkeyEncryption	SecurityLevel	KeyType

Figure 117 – Format of the Options field of GPD Commissioning Reply command

The *PAN ID present* sub-field, if set to 0b1, indicates that the *PANId* field is present, and carries the value of the network operational PANId.

When the *GPDsecurityKeyPresent* sub-field is set to 0b1 and the *GPDkeyEncryption* sub-field is set to 0b0, then the *GPDkeyMIC* field is absent, and the *SecurityKey* field is present in the clear, and carries the key type as indicated in the *KeyType* field of the *Options* field. When the *GPDsecurityKeyPresent* sub-field is set to 0b1 and the *GPDKeyEncryption* sub-field is set to 0b1, then both fields *GPDsecurityKey* and *GPDkeyMIC* are present; the field *GPDsecurityKey* contains the *gpdSecurityKey*, of the type as indicated in the *KeyType* sub-field, encrypted with the default TC-LK (see A.3.3.3.3) as described in A.3.7.1.2.3; and the *GPDkeyMIC* field contains the MIC for the encrypted GPD key, calculated as described in A.3.7.1.2.3. When the *GPDsecurityKeyPresent* sub-field is set to 0b0, the *GPDKeyEncryption* sub-field is ignored.

If the *SecurityLevel* sub-field is set to 0b00, the *GPDsecurityKey* field is not present and the sub-fields *GPDkeyEncryption* and *KeyType* SHALL be set to 0b0 and 0b000, respectively, on transmission and ignored upon reception.

The *SecurityLevel* sub-field indicates the requested gpdSecurityLevel.

The *KeyType* sub-field contains the type of the key to be used for GPDF protection in operation, and can take values as defined in Table 53.

The *Frame Counter* field is only present when the sub-fields of the *Options* field are set as follows: *SecurityLevel* sub-field to 0b10 or 0b11, *GPDsecurityKeyPresent* sub-field to 0b1 and the *GPDkeyEncryption* sub-field to 0b1; otherwise it is absent. It carries the security frame counter value that was used to encrypt the shared security key transmitted (see A.3.7.1.2.3).

A.4.2.1.2.2 When generated

The GPD Commissioning Reply command is generated by the commissioning sink upon receipt of a GPD Commissioning command with the *RxAfterTx* sub-field set to 0b1, if all application requirements on the GPD capabilities are met (see sec. A.3.6.2.1).

A.4.2.1.2.3 Effect on receipt

On receipt of this Commissioning Reply GPDF, the GPD checks if the *GPD SrcID*/IEEE address field value matches its own identifier. If not, it SHALL drop this frame. If the GPD is the destination of this Commissioning Reply GPDF, and the security check succeeds, the GPD SHALL update all the requested parameters with the values present in the frame payload. The GPD SHALL only reset its security frame counter to 0x00000000 if upon GPD Commissioning Reply command reception the security frame counter of the GPD is larger than 0x80000000 AND the type or value of the supplied key differs from the key currently used.

The GPD MAY support GPD Commissioning Reply command in operational mode.

A.4.2.1.3 Decommissioning command

The GPD Decommissioning command does not have any payload.

A.4.2.1.3.1 When generated

The Decommissioning GPDF is sent by the GPD to initiate its removal from the network. The Decommissioning GPDF SHALL be sent protected, if the GPD supports security.

A.4.2.1.3.2 Effect on receipt

On reception of GPD Decommissioning command, the proxies act as described in A.3.5.2.1, and the sinks act as described in A.3.5.2.4.

A.4.2.1.4 Channel Request command

The payload of the Channel Request command is formatted as shown in Figure 118.

Octets	**1**
Data Type	8-bit bitmap
Field name	*Channel toggling behavior*

Figure 118 – Format of the GPD Channel Request command payload

The *Channel Toggling Behavior* field is formatted as shown in Figure 119.

Bits: 0-3	**4-7**
Rx channel in the next attempt	Rx channel in the second next attempt

Figure 119 – Format of the Channel Toggling Behavior field of the GPD Channel Request command

The *Rx channel in the (second) next attempt* sub-field can take the following values: 0b0000: channel 11, 0b0001: channel 12, …, 0b1111: channel 26.

The Channel Request GPDF can use the following values of the *Frame Type* sub-field of the *NWK Frame Control* field: 0b01 and 0b00.

When sent as part of the commissioning procedure, the GPD Channel Request command SHALL be sent with *Frame Type* sub-field of the *NWK Frame Control* field set to 0b01 (Maintenance frame; see sec. A.1.4.1.2).

When sent in operational mode, the GPD Channel Request command SHALL be sent with *Frame Type* sub-field of the *NWK Frame Control* field set to 0b00 (Data frame; see sec. A.1.4.1.2); it SHALL then be secured with the security settings as established during the commissioning.

A.4.2.1.5 Channel Configuration command

The payload of the Channel Configuration command is formatted as shown in Figure 120.

Octets	**1**
Data Type	8-bit bitmap
Field name	*Channel*

Figure 120 – Format of the GPD Channel Configuration command payload

The *Channel* field is formatted as shown in Figure 121.

Bits: 0-3	4	5-7
Operational Channel	Basic	Reserved

Figure 121 – Format of the Channel field of the GPD Channel Configuration command

The *OperationalChannel* sub-field can take the following values: 0b0000: channel 11, 0b0001: channel 12, …, 0b1111: channel 26.

The *Basic* sub-field indicates if the sender is a basic only GP infrastructure device or if it supports bidirectional operation. This bit SHALL be set to 0b1 in GPD Channel Configuration commands sent by Basic Combo product.
The Channel Configuration GPDF can use the following values of the *Frame Type* sub-field of the *NWK Frame Control* field: 0b01 and 0b00.

When sent as part of the commissioning procedure, the GPD Channel Configuration command SHALL be sent with *Frame Type* sub-field of the *NWK Frame Control* field set to 0b01 (Maintenance frame; see sec. A.1.4.1.2).

When sent in operational mode, the GPD Channel [288]Configuration command SHALL be sent with *Frame Type* sub-field of the *NWK Frame Control* field set to 0b00 (Data frame; see sec. A.1.4.1.2); it SHALL then be secured with the security settings as established during the commissioning.

A.4.2.1.6 Application Description command

The command payload for the GPD Application Description command is formatted as shown in Figure 122.

Octets	1	1	Variable	…	Variable
Data Type	Unsigned 8-bit integer	Unsigned 8-bit integer	Sequence of unsigned 8-bit integer	…	Sequence of unsigned 8-bit integer
Field name	Total number of reports	Number of reports	Report descriptor M	…	Report descriptor N

Figure 122 – Payload of the GPD Application Description command

[289]The *Total number of reports* field carries the total number of different *Report descriptors* this GPD will be sending during the commissioning process; they may be spread across multiple GPD Application Description commands. [290]The *Total number of reports* field SHALL be set to a value other than 0x00.

[291]The *Number of reports* field carries the number of the *Report descriptor* fields present in the current GPD Application Description command. [292]The *Number of reports* field SHALL be set to a value other than 0x00 and smaller than [293]or equal to the value in the *Total number of reports*.

A *Report descriptor* field defined the layout of one GPD Compact Attribute Reporting command that this GPD supports. The *Report descriptor* is formatted as shown in Figure 123.

[288] CCB #2361; Resolution added in GP Basic spec errata 15-02014-011
[289] Comment #9 from GP multi-sensor August PoC, Zigbee document 16-02611
[290] Comment #8 from GP multi-sensor August PoC, Zigbee document 16-02611
[291] Comment #9 from GP multi-sensor August PoC, Zigbee document 16-02611
[292] Comment #8 from GP multi-sensor August PoC, Zigbee document 16-02611
[293] Comment #772 from GP multi-sensor v0.7 letter ballot

Octets	1	1	0/2	1	Variable	…	Variable
Data Type	Unsigned 8-bit integer	8-bit bitmap	Unsigned 16-bit integer	Unsigned 8-bit integer	Sequence of unsigned 8-bit integer	…	Sequence of unsigned 8-bit integer
Field name	Report identifier	Report Options	Timeout period	Remaining length of report descriptor	Data point descriptor 1	…	Data point descriptor N

Figure 123 – Format of the *Report descriptor* field of the GPD Application Description command

The *Report identifier* field carries the index value for the report being described. [294]The lowest report SHALL have the *Report identifier* value of 0, and the other reports SHALL use consecutive numbers for the *Report identifier* value up to *Total number of reports* - 1.

The *Report Options* field is formatted as shown in Figure 124.

Bits: 0	1..7
Timeout period present	Reserved

Figure 124 – Format of the *Report Options* field of the Report descriptor fields of the GPD Application Description command

The *Timeout period present* sub-field is a Boolean flag. If set to 0b1, it indicates that the *Timeout period* field is present. If set to 0b0, it indicates that the *Timeout period* field is absent.

The *Timeout period* field, if present, carries the maximum time duration, in seconds, between the consecutive reports with the same *Report identifier*. A GPD SHALL only include this value if reporting intervals for a particular *Report identifier* are fixed or a maximum interval is defined. A GP infrastructure device MAY start some maintenance actions, e.g. if no report is received since a multiple of the *Timeout period*; any such actions are out of scope of the current specification.

The *Remaining length of report descriptor* field carries the total number, in octets, of all the following *Data point descriptor* fields belonging to the current report descriptor. The *Remaining length of report descriptor* field indicates to the sink where the current report descriptor ends.

The *Data point descriptor* field is formatted as shown in Figure 125.

Octets	1	2	0/2	Variable	…	Variable
Data Type	8-bit bitmap	16-bit enumeration	16-bit enumeration	Sequence of unsigned 8-bit integer	…	Sequence of unsigned 8-bit integer
Field name	Data point options	ClusterID	ManufacturerID	Attribute record 1	…	Attribute record N

Figure 125 – Format of the *Data point descriptor* field of the GPD Application Description command

The *Data point options* field is formatted as shown in Figure 126.

Bits: 0..2	3	4	5..7
Number of attribute records	Client / server	ManufacturerID present	Reserved

Figure 126 – Format of the *Data point options* field of the *Data point descriptor* fields of the GPD Application Description command

[294] Comment #777 from GP multi-sensor v0.7 letter ballot

The *Number of attribute records* sub-field of the *Data point options* field carries the number of *Attribute record* fields that follow, decremented by 1. Thus, *Number of attribute records* = 0b000 indicates that one *Attribute record* field follows; *Number of attribute records* = 0b111 indicates that eight *Attribute record* fields follow.

The *Client / server* sub-field is a Boolean flag. If set to 0b1, it indicates the GPD implements the server side of the cluster identified by the *ClusterID* field. If set to 0b0, it indicates the GPD implements the client side of the cluster identified by the *ClusterID* field.

The *ManufacturerID present* sub-field is a Boolean flag. If set to 0b1, it indicates that the *ManufacturerID* field is present. If the *ClusterID* is from a manufacturer-specific range, as defined in the Zigbee ZCL [3], or if the [295]*AttributeID* is from the Green Power manufacturer-specific attribute range, as defined in Table 58, the attribute is manufacturer-specific; otherwise the attribute as indicated by the *AttributeID* field is a standard attribute of the cluster identified by *ClusterID* as defined in the ZCL [3].

ClusterID field carries the value of the ClusterID as defined in the public Zigbee ZCL [3].

The *Attribute record* field is formatted as shown in Figure 127.

Octets	2	1	1	0/1	0/Variable
Data Type	16-bit integer	8-bit enumeration	8-bit bitmap	8-bit integer	variable
Field name	Attribute ID	Attribute Data Type	Attribute Options	Attribute Offset within Report	Attribute value

Figure 127 – Format of the *Attribute record* field of the GPD Application Description command

The *Attribute ID* field carries the value of the AttributeID of the cluster indicated in the *ClusterID* field as defined in the public Zigbee ZCL [3]. The standard and manufacturer-specific attributes SHALL use appropriate AttributeIDs, as defined in Table 58.

The *Attribute Data Type* field carries the data type of the attribute to be reported.

The *Attribute Options* field is formatted as shown in Figure 128.

Bits: 0..3	4	[296]5	6..7
Remaining Attribute Record Length	Reported	Attribute value present	Reserved

Figure 128 – Format of the *Attribute Options* field of the Attribute record fields of the GPD Application Description command

The *Remaining Attribute Record Length* field carries the total number in octets decremented by one, of the following *Attribute record* fields. Thus, *Remaining Attribute Record Length* = 0b000 indicates that one octet follows, etc. The *Remaining Attribute Record Length* field allows the sink for skipping *Attribute records* for *AttributeIDs* it does not support.

[295] Comment #781 from GP multi-sensor v0.7 letter ballot
[296] Comment #8 from GP multi-sensor August PoC, Zigbee document 16-02611

The *Reported* sub-field is a Boolean flag which indicates if the attribute as identified by the AttributeID field is reported by the GPD in operation, or if it is background data required for processing of a reported attribute only conveyed once at commissioning time. [297]For example, if a GPD implements the server side of the Temperature Measurement cluster, it will include in the GPD Application Description command the reportable *MeasuredValue* attribute, and it can include as non-reportable any of the other, static attributes of the Temperature Measurement cluster: *MinMeasuredValue*, *MaxMeasuredValue* and *Tolerance*. If *Reported* = 0b1, *Attribute Offset within Report* field is present, [298]otherwise it is absen~~tand the *Attribute value* field is absent. If *Reported* = 0b0, *Attribute Offset within Report* field is absent and the *Attribute value* field is present~~.

[299]The *Attribute value present* sub-field is a Boolean flag. If *Attribute value present* = 0b1, the *Attribute value* field is present; otherwise it is absent. Note: since the Application Description GPDF is sent unprotected, including the *Attribute value* may not always be desired.

[300]At least one of the sub-fields *Reported* and *Attribute value present* SHALL be set to 0b1.

The *Attribute Offset within Report* field, when present, carries the start position (in bytes) of the data point identified by the *AttributeID* of the *ClusterID* in the report payload. The *Attribute Offset within Report* = 0x00 corresponds to the [301]octet immediately following the *Report identifier* field in the payload of the GPD Compact Attribute Reporting command.

The *Attribute value* field, when present, carries the actual fixed value of that attribute; *the length and type of this field are determined by the *AttributeID* of the *ClusterID* (in case of manufacturer-specific attributes or clusters, corresponding to the *ManufacturerID*).

A.4.2.2 Generic switch commands

The advanced generic switch GPD determines is the switch operation was a short or long press. The time threshold to determine short or long press duration is implementation-specific. The recommended value is 300ms.

A.4.2.2.1 GPD 8-bit vector: press/release

The payload of the commands GPD 8-bit vector: press and GPD 8-bit vector: release is formatted as shown in Figure 120.

Octets	1
Data Type	8-bit bitmap
Field name	Contact status

Figure 129 – Format of the GPD Press: 8-bit vector and Release: 8-bit vector command payload

The *Contact status* field is an 8-bit bitmap. Only N least significant bits SHALL be processed, where N is the value as indicated in the *Number of contacts* sub-fields of the *Generic switch configuration* field of the GPD Commissioning command. The remaining bits SHALL be set to 0b0 upon transmission and ignored upon reception.

The values of the individual sub-fields of the *Contact status* field have the following meaning for both the GPD 8-bit vector: press command and the GPD 8-bit vector: release command: a sub-field set to:

- 0b1 indicates a closed contact;

[297] Comment #782 from GP multi-sensor v0.7 letter ballot
[298] Comment #8 from GP multi-sensor August PoC, Zigbee document 16-02611
[299] Comment #8 from GP multi-sensor August PoC, Zigbee document 16-02611
[300] Comment #8 from GP multi-sensor August PoC, Zigbee document 16-02611
[301] Comment #10 from GP multi-sensor August PoC, Zigbee document 16-02611

- 0b0 indicates an open contact.

For a rocker switch – either pre-configured, as indicated in the *Switch type* sub-field of the *Generic switch configuration* field of the GPD Commissioning command or a generic switch which can be configured as a rocker [302]by applying actuation elements of appropriate mechanical design - the contacts triggered by the same rocker SHALL be represented on consecutive bits of the *Contact status* vector, occupying the same 2-bit nibble, starting from the least significant bit of the vector, i.a. b0-b1, b2-b3, etc.).

The 2-bit nibble SHOULD be used as follows:

- The lower (even) bit to represent off or (dim) down side of the rocker;
- The higher (odd) bit to represent on or (dim) up side of the rocker.

For example, on a rocker using the b0-b1 nibble, b0 represents off and b1 represents on.

For other switch types, the supported contacts SHOULD be mapped in increasing order on the least significant bits of the *Contact status* field, i.e. contact 1 on b0, etc.

[303]A GPD supporting generic switch functionality (GPD CommandID 0x69 and/or 0x6a) SHALL be capable of subsequent commissioning, i.e. performing the commissioning procedure sequentially for each supported button without prior reset.

A.4.2.3 Sensor commands

All sensor commands defined in this section SHALL be used with *Auto-Commissioning* sub-field of the *NWK Frame control* field set to 0b0. I.e. all devices implementing the sensor commands SHALL be capable of sending GPD Commissioning command (see sec. A.4.2.1.1).

A sink supporting GPD sensor functionality SHALL support all sensor commands defined in this section.

GPD sensors and GPDs supporting sensor functionality SHALL support at least one sensor command defined in this section.

If GPD command 0xA6 is supported, and bidirectional operation is supported, the GPD command 0xF6 SHALL be supported as well; this applies to both GPDs and sinks.

If a ZCL command carried in 0xA6 or 0xF6 command requires a response, the response SHALL be sent using the 0xF6 or 0xA6 command, respectively.

To yet better accommodate for energy-efficient exchange of information on multiple attributes in one GPD command, the current specification defines a manufacturer-specific attribute range, see Table 58. This attribute range definition applies to the sensor commands specified in the current section, as well as to the bidirectional operation commands in sec. A.4.2.6.

Table 58 – Attribute ranges for GPD commands

Value	Description
0x0000 – 0x4fff	ZCL defined public attribute range
0x5000 – 0xffff	Recommended manufacturer-specific attribute range

The GPD commands containing attributes from the manufacturer-specific range SHALL also contain ManufacturerID. If ManufacturerID is not present those AttributeIDs SHALL NOT be processed.

[302] LB v07: https://workspace.zigbee.org/kws/groups/zigbee_pro_foundation/comments/view_comment?comment_id=280
[303] Generic switch commissioning guidelines, Zigbee document 16-02604-004
Dec 2016 SVE comment: https://workspace.zigbee.org/kws/groups/PRO_GP/comments/view_comment?comment_id=1012

If ManufacturerID field is included in any of the GPD commands in this section, any commands of standard ClusterIDs or attributes of standard ClusterIDs from the ZCL-defined public range SHALL be interpreted as standard commands and attributes as defined in the ZCL [3], irrespective of this ManufacturerID being supported or not. All attributes of manufacturer-specific ClusterIDs and attributes of standard ClusterIDs from the manufacturer-specific range SHALL be interpreted in the context of the ManufacturerID.

A.4.2.3.1 Attribute Reporting command

The command payload for the GPD Attribute Reporting command is formatted as shown in Figure 130.

Octets	**2**	**variable**	**variable**	**…**	**variable**
Data Type	Unsigned 16-bit integer	structure	structure	…	structure
Field name	Zigbee Cluster ID	Attribute report 1	Attribute report 2	…	Attribute report n

Figure 130 – Payload of the GPD Attribute Reporting command

Zigbee Cluster ID field carries the value of the ClusterID defined in the public Zigbee ZCL which attributes are reported by the GPD sensor. For example, if the GP sensor reports temperature attributes, the Public Zigbee ClusterID is set to value *0x0402* which is the Temperature measurement cluster ID defined in the ZCL.

Attribute report field SHALL be formatted as depicted in Figure 131.

Octets	**2**	**1**	**variable**
Field name	AttributeID	Attribute data type	Attribute data

Figure 131 – Format of the *Attribute report* field

AttributeID field is 16-bits in length and SHALL contain the identifier of the attribute that is being reported.

Attribute Data Type field contains the data type of the attribute that is being reported.

Attribute Data field is variable in length and SHALL contain the actual value of the attribute being reported.

There is no limit on the number of attributes reported in a single Attribute Reporting command.

A.4.2.3.2 Manufacturer-Specific Attribute Reporting command

The command payload for the GPD Manufacturer-Specific Attribute Reporting command is formatted as shown in Figure 132.

Octets	**2**	**2**	**variable**	**variable**	**…**	**variable**
Data Type	Unsigned 16-bit integer	Unsigned 16-bit integer	structure	structure	…	structure
Field name	Manufacturer Code	Cluster ID	Attribute report 1	Attribute report 2	…	Attribute report n

Figure 132 – Payload of the GPD Manufacturer-Specific Attribute Reporting command

Manufacturer Code field SHALL be set to the value of the manufacturer ID. It can take values as defined in [7].

ClusterID field SHALL have the value of the cluster ID defined by the manufacturer which attributes are reported by the GPD sensor.

Attribute report field SHALL be formatted as depicted in Figure 131.

A.4.2.3.3 Multi-Cluster Reporting command

The command payload for the GPD Multi-cluster reporting command is formatted as shown in Figure 133.

Octets	**variable**	**variable**	**…**	**variable**
Data Type	structure	structure	…	structure
Field name	Cluster report 1	Cluster report 2	…	Cluster report n

Figure 133 – Payload of the GPD Multi-Cluster Reporting command

Cluster report field SHALL be formatted as depicted in Figure 134.

Octets	**2**	**2**	**1**	**variable**
Field name	ClusterID	AttributeID	Attribute data type	Attribute data

Figure 134 – Format of the *Cluster report* field

ClusterID field carries the value of the ClusterID defined in the public Zigbee ZCL which attributes are reported by the GPD sensor.

AttributeID field is 16-bits in length and SHALL contain the identifier of the attribute that is being reported.

Attribute Data Type field contains the data type of the attribute that is being reported.

Attribute Data field is variable in length and SHALL contain the actual value of the attribute being reported.

There is no limit on the number *of cluster report* fields reported in a single Multi-Cluster Reporting command.

If a GPD has multiple attributes of the same cluster to report, it is recommended to put them one after the other, so that the receiving sink can aggregate them in the same ZCL message to the sink's local application endpoint.

A.4.2.3.4 Manufacturer-Specific Multi-Cluster Reporting command

The command payload for the GPD Manufacturer-Specific Multi-Cluster Reporting command is formatted as shown in Figure 135.

Octets	**2**	**variable**	**variable**	**…**	**variable**
Data Type	Unsigned 16-bit integer	structure	structure	…	structure
Field name	Manufacturer Code	Cluster report 1	Cluster report 2	…	Cluster report n

Figure 135 – Payload of the GPD Manufacturer-Specific Multi-Cluster Reporting command

The *Manufacturer Code* carries the Manufacturer ID. It can take values as defined in [7].

Cluster report field SHALL be formatted as depicted in Figure 134. The ClusterID carries the cluster identified as defined by the manufacturer.

There is no limit on the number of *cluster report* fields reported in a single Manufacturer-Specific Multi-Cluster Reporting command.

If a GPD has multiple attributes of the same cluster to report, it is recommended to put them one after the other, so that the receiving sink can aggregate them in the same ZCL message to the sink's local application endpoint.

A.4.2.3.5 GPD ZCL Tunneling commands

The GPD supporting the transmission of GPD ZCL Tunneling command (0xA6) SHALL at least support the tunneled ZCL functionality equivalent to the GPD functionality mandated for this particular GPD DeviceID (see [13]).

The GPD supporting the reception of GPD ZCL Tunneling command (0xF6) SHALL at least support the tunneled ZCL functionality equivalent to the GPD functionality mandated for this particular GPD DeviceID (see [13]).

GPD MAY in addition support tunneling of other ZCL functionality.

If the GPD supports GPD ZCL Tunneling for ZCL-defined clusters not referenced by the GPD specification (see [13]), it SHALL support all the functionality mandated by the ZCL (see [3]) for this cluster.

For the received ZCL Tunneling command (0xF6), the GPD SHALL process all attributes and commands that are implemented. If a response is required, the GPD SHALL send it with the appropriate Status value, if required; the GPD MAY choose to send multiple responses. If the received ZCL Tunneling command references any clusters, commands or attributes not supported by the GPD, the GPD MAY respond with a corresponding commands with the Status UNSUPPORTED_ATTRIBUTE (for the values of the Status codes see [3]).

This section defines the payload of both GPD ZCL Tunneling commands, 0xA6 and 0xF6.

The command payload for the ZCL Tunneling command is formatted as shown in Figure 136.

Octets	**1**	**0/2**	**2**	**1**	**1**	**0/Variable**
Data Type	8-bit bitmap	16-bit enumeration	Unsigned 16-bit integer	unsigned 8-bit integer	unsigned 8-bit integer	Sequence of unsigned 8-bit integer
Field name	Options	ManufacturerID	Zigbee Cluster ID	Zigbee Command ID	Length of Payload	Zigbee Command Payload

Figure 136 – Payload of the GPD ZCL Tunneling command

The *Options* field is formatted as shown in Figure 137.

Bits: 0-1	**2**	**3**	**4..7**
Frame type	ManufacturerID present	Direction	Reserved

Figure 137 – Format of the Options field of the GPD ZCL Tunneling command

The *Frame type* sub-field specifies the frame type of the ZCL command (cluster-specific or ZCL generic), as defined in section 2.3.1.1.1 of the [3].

The *ManufacturerID present* sub-field defines if the ZCL Tunneling command is for standard clusters or manufacturer specific clusters. The *ManufacturerID* field can take values as defined in [7]. If the *ManufacturerID present* sub-field is set to 0b0, the *ManufacturerID* field SHALL be omitted; the *Zigbee ClusterID* field contains standard Zigbee Cluster ID. If the *ManufacturerID present* sub-field is set to 0b1, the *ManufacturerID* field SHALL be present; the following *ClusterID* field contains a manufacturer-specific cluster corresponding to the *ManufacturerID*.

The *Direction* sub-field defines the client-server direction of the content carries by the ZCL Tunneling command. It takes the values as defined in section 2.3.1.1.3 of the ZCL [3].

Zigbee Cluster ID field carries the value of the ClusterID. The *Zigbee Cluster ID* field can take values as defined in section 2.5.1.3 of [3].

Zigbee Command ID field carries the value of the Zigbee Command ID, either cluster-specific command of the specified *Zigbee ClusterID* or generic ZCL command as defined in section 2.4 of [3].

Length of Payload field carries the length of the *Zigbee Command Payload* field in octets.

Zigbee Command Payload field carries the ZCL frame payload specific for the *Zigbee Command ID*.

A.4.2.3.6 Compact Attribute Reporting command

The command payload for the GPD Compact Attribute Reporting command is formatted as shown in Figure 138.

Octets	1	Variable	…	Variable
Data Type	Unsigned 8-bit integer	Variable	…	Variable
Field name	Report identifier	Data point 1	…	Data point N

Figure 138 – Payload the GPD Compact Attribute Reporting command

The *Report identifier* field carries the pointer to the current report structure, as indicated before in the GPD Application Description command (see sec. A.4.2.1.6).

Each data point is of length and type as indicated before in the GPD Application Description command for this *Report identifier* value.

The data points currently reportable using the Compact Attribute Reporting mechanism are listed in [13].

A.4.2.4 Level control commands

A.4.2.4.1 Move Up

The command payload for the Move Up command is modelled after the Move command of the ZCL Level Control Cluster and is formatted as shown in Figure 139.

Octets	0/1
Data Type	Unsigned 8-bit integer
Field name	Rate

Figure 139 – Payload the GPD Move Up command

The *Rate* field specifies the rate of movement in units per second. The actual rate of movement SHOULD be as close to this rate as the device is able. [304]If the device is not able to move at a variable rate, this field MAY be disregarded.

[304] PoC comment #2, #3 (Zigbee document 16-02601)

[305]The presence of the *Rate* field is optional, and can be deduced from the command payload length. If the *Rate* field is not present or if it is present but set to 0xff, indicating unspecified, then the receiver SHALL move at an implementation-specific default rate, if it has a variable rate, or else at the only available rate.
Note: Is the default rate is very high, the execution of the GPD Move Up command may appear to the user identical to execution of a GPD On command

A.4.2.4.2 Move Down

The command payload for the Move Down command is modelled after the Move command of the ZCL Level Control Cluster and is formatted as shown in Figure 139.

[306]The *Rate* field is defined in sec. A.4.2.4.1.

A.4.2.4.3 Step Up

The command payload for the Step Up command is modelled after the Step command of the ZCL Level Control Cluster and is formatted as shown in Figure 140.

Octets	1	0/2
Data Type	Unsigned 8-bit integer	Unsigned 16-bit integer
Field name	Step size	Transition time

Figure 140 – Payload the GPD Step Up command

The *Transition time* field specifies the time that SHALL be taken to perform the step, in tenths of a second. A step is a change in the *CurrentLevel* of 'Step size' units. The actual time taken SHOULD be as close to this as the device is able. [307]If the device is not able to move at a variable rate, the Transition time field MAY be disregarded.

The presence of the *Transition time* field is optional, and can be deduced from the command payload length. [308]If the *Transition time* field is not present, or if it is present but set to 0xffff, indicating unspecified then the receiver SHALL move at an implementation-specific default rate, if it has a variable rate, or else at the only available rate.

A.4.2.4.4 Step Down

The command payload for the Step Down command is modelled after the Step command of the ZCL Level Control Cluster and is formatted as shown in Figure 140.

[309]The payload fields are defined in sec.A.4.2.4.4.

A.4.2.4.5 'With On/Off' Commands

[310]The Move Up/Down (with On/Off) and Step Up/Down (with On/Off) commands have identical payloads to the Move Up/Down commands (see sec. A.4.2.4.1) and Step Up/Down commands (see sec. A.4.2.4.3), respectively.

They also have the same effects on reception, except for the following additions.

- Before commencing any command that has the effect of increasing *CurrentLevel*, the *OnOff* attribute of the On/Off cluster on the same endpoint, if implemented, SHALL be set to On.

[305] PoC comment #2, #3 (Zigbee document 16-02601)
[306] PoC comment #2, #3 (Zigbee document 16-02601)
[307] PoC comment #2, #3 (Zigbee document 16-02601)
[308] PoC comment #2, #3 (Zigbee document 16-02601)
[309] PoC comment #2, #3 (Zigbee document 16-02601)
[310] PoC comment #2, #3 (Zigbee document 16-02601)

- If any command that decreases *CurrentLevel* reduces it to the minimum level allowed by the device, the *OnOff* attribute of the On/Off cluster on the same endpoint, if implemented, SHALL be set to Off.

A.4.2.5 [311]Color control

A.4.2.5.1 Move Hue Up/Down

The command payload for the Move Hue Up/Down command is modelled after the Move Hue command of the ZCL Color Control Cluster and is formatted as shown in Figure 139.

The *Rate* field specifies the rate of movement in steps per second. A step is a change in the device's hue of one unit. If the *Rate* field has a value of zero, the command has no effect; no ZCL default response command SHALL be sent.

The presence of the *Rate* field is optional, and can be deduced from the command payload length. If the *Rate* field is not present, or if it is present but set to 0xff, indicating unspecified, then the receiver SHALL move at an implementation-specific default rate, if it has a variable rate, or else at the only available rate.

A.4.2.5.2 Step Hue Up/Down

The command payload for the Step Hue Up/Down command is modelled after the Step Hue command of the ZCL Color Control Cluster and is formatted as shown in Figure 140.

The *Transition time* field specifies, in 1/10ths of a second, the time that SHALL be taken to perform a single step. A step is a change in the device's hue of '*Step size*' units. Note that if the color specified is not achievable by this hardware then the color SHALL NOT be set and no ZCL default response command SHALL be generated.

The presence of the *Transition time* field is optional, and can be deduced from the command payload length. If the *Transition time* field is not present, or if it is present but set to 0xffff, indicating unspecified then the receiver SHALL move at an implementation-specific default rate, if it has a variable rate, or else at the only available rate.

A.4.2.5.3 Move Saturation Up/Down

The command payload for the Move Saturation Up/Down command is modelled after the Move Saturation command of the ZCL Color Control Cluster and is formatted as shown in Figure 139.

The *Rate* field specifies the rate of movement in steps per second. A step is a change in the device's saturation of one unit. If the *Rate* field has a value of zero, the command has no effect; no ZCL default response command SHALL be sent.

The presence of the *Rate* field is optional, and can be deduced from the command payload length. If the *Rate* field is not present, or if it is present but set to 0xff, indicating unspecified, then the receiver SHALL move at an implementation-specific default rate, if it has a variable rate, or else at the only available rate.

A.4.2.5.4 Step Saturation Up/Down

The command payload for the Step Saturation Up/Down command is modelled after the Step Saturation command of the ZCL Color Control Cluster and is formatted as shown in Figure 140.

[311] PoC comment #2, #3 (Zigbee document 16-02601)

The *Transition time* field specifies, in 1/10ths of a second, the time that SHALL be taken to perform a single step. A step is a change in the device's saturation of '*Step size*' units. Note that if the color specified is not achievable by this hardware then the color SHALL NOT be set and no ZCL default response command SHALL be generated.

The presence of the *Transition time* field is optional, and can be deduced from the command payload length. If the *Transition time* field is not present, or if it is present but set to 0xffff, indicating unspecified then the receiver SHALL move at an implementation-specific default rate, if it has a variable rate, or else at the only available rate.

A.4.2.5.5 Move Color

The command payload for the Move Color command is modelled after the Move Color command of the ZCL Color Control Cluster and is formatted as shown in Figure 141.

Octets	**2**	**2**
Data Type	Signed 16-bit integer	Signed 16-bit integer
Field name	RateX	RateY

Figure 141 – Payload of the GPD Move Color command

The *RateX* field specifies the rate of movement in steps per second. A step is a change in the device's *CurrentX* attribute of one unit. The *RateY* field specifies the rate of movement in steps per second. A step is a change in the device's *CurrentY* attribute of one unit. This movement SHALL continue until either the new color cannot be implemented on this device, or this command is received with the RateX and RateY fields both containing a value of zero.

A.4.2.5.6 Step Color

The command payload for the Step Color command is modelled after the Step Color command of the ZCL Color Control Cluster and is formatted as shown in Figure 142.

Octets	**2**	**2**	**0/2**
Data Type	Signed 16-bit integer	Signed 16-bit integer	Unsigned 16-bit integer
Field name	StepX	StepY	Transition time

Figure 142 – Payload the GPD Step Color command

The *StepX* and *StepY* fields specify the change to be added to the device's *CurrentX* attribute and *CurrentY* attribute respectively. The *Transition time* field specifies, in 1/10ths of a second, the time that SHALL be taken to perform the color change.

The presence of the *Transition time* field is optional, and can be deduced from the command payload length. If the *Transition time* field is not present, or if it is present but set to 0xffff, indicating unspecified, then the receiver SHALL move at an implementation-specific default rate, if it has a variable rate, or else at the only available rate.

A.4.2.6 Bidirectional operation commands

A.4.2.6.1 Request Attributes command

The command payload of the Request Attributes command is formatted as shown in Figure 143.

Octets	1	0/2	variable	…	variable
Data Type	8-bit bitmap	Unsigned 16-bit integer	Structure	…	structure
Field name	Options	Manufacturer ID	Cluster Record Request	…	Cluster Record Request

Figure 143 – Payload of the GPD Request Attributes command

The *Options* field is formatted as shown in Figure 144.

Bits: 0	1	2..7
Multi-record	Manufacturer field present	Reserved

Figure 144 – Format of the Options field of the GPD Request Attributes command

The Multi-Record sub-field, if set to 0b1, indicates that the Request Attributes command carries multiple *Cluster Record Request* fields. If set to 0b0, the Request Attributes command contains a single *Cluster Record Request*.

The *Manufacturer field present* sub-field defines if the Request Attributes command is for standard clusters or manufacturer specific clusters. If the *Manufacturer field present* sub-field is set to 0b0, the *ManufacturerID* field SHALL be omitted; all the following *ClusterID* fields in the *Cluster Record Requests* in this command contain standard Zigbee Cluster IDs. If the *Manufacturer field present* sub-field is set to 0b1, the *ManufacturerID* field SHALL be present; all the following *ClusterID* fields in the *Cluster Record Requests* in this command contain manufacturer-specific cluster corresponding to the *ManufacturerID*. The *ManufacturerID* field can take values as defined in [7].

The *Cluster Record Request* field is formatted as shown in Figure 145. Each *Cluster Record Request* allows for requesting the value of one or multiple *Attributes* belonging to one particular cluster, as identified in the *ClusterID* field.

Octets	2	1	2	…	2
Data Type	Unsigned 16-bit integer	Unsigned 8-bit integer	Unsigned 16-bit integer	…	Unsigned 16-bit integer
Field name	Cluster ID	*Length of Record List*	Attribute	…	Attribute

Figure 145 – Format of the Cluster Record Request field

The *Length of Record List* field indicates the total size in octets of the following *Attribute* list until the next *ClusterID* field.

A.4.2.6.2 Read Attributes Response command

The Read Attributes Response command is sent by the GPD in response to the Read Attributes command. The GPD SHALL send Read Attributes Response command with the *Status* SUCCESS for all requested attributes that are implemented; the GPD MAY send one or multiple Read Attribute Response commands, as required.

For attributes contained in the Read Attributes Request not supported by the GPD, the GPD MAY send one or multiple Read Attributes Response commands with *Status* UNSUPPORTED_ATTRIBUTE. If *ManufacturerID* field is included, all attributes in *Cluster record* fields with standard *ClusterID* contained in the Read Attributes command SHALL be interpreted as standard attributes defined in the ZCL [3]. Read Attributes Response SHALL be created for those attributes, if implemented, irrespective of this *ManufacturerID* being supported or not. All attributes in *Cluster record* fields with manufacturer-specific *ClusterIDs* SHALL be interpreted in the context of the *ManufacturerID*; one or multiple Read Attributes Response SHALL be sent with *Status* SUCCESS if *ManufacturerID*, manufacturer-specific *ClusterID* and a particular attribute are implemented; otherwise, Read Attribute Response with *Status* UNSUPPORTED_ATTRIBUTE MAY be returned.

The command payload for the Read Attributes Response command is formatted as shown in Figure 146.

Octets	**1**	**0/2**	**variable**	**…**	**variable**
Data Type	8-bit bitmap	Unsigned 16-bit integer	structure	…	structure
Field name	Options	Manufacturer ID	Cluster record	…	Cluster record

Figure 146 – Payload of the GPD Read Attributes Response command

The *Options* field is formatted as shown in Figure 144, and the sub-fields are defined as in A.4.2.6.1.

The *Manufacturer ID* field can take values as defined in [7].

The *Cluster record* field is formatted as shown in Figure 147.

2	**1**	**variable**	**variable**	**…**	**variable**
Unsigned 16-bit integer	Unsigned 8-bit integer	structure	structure	…	structure
Cluster ID	Length of record list	Read Attribute record	Read Attribute record	…	Read Attribute record

Figure 147 – Format of the Cluster record field

The *Length of Record List* field indicates the total size in octets of the following Read Attribute Record list until the next Cluster ID field. The *Read Attribute Record* field is formatted as shown in Figure 148.

The *Status* field specifies the status of the read operation on this attribute. This field SHALL be set to SUCCESS, if the operation was successful, or an error code, as specified in Table 2.16 of [3], if the operation was not successful.

Octet: 2	**1**	**1**	**Variable**
Unsigned 16-bit integer	8-bit enumeration	8-bit enumeration	variable
AttributeID	Status	Attribute Data Type	Attribute Value

Figure 148 – Format of the Read attribute record field

If the *Manufacturer field present* sub-field is set to 0b0, all the *ClusterID* fields in the *Attribute Record* fields of this command contain standard Zigbee Cluster IDs, with attributes as defined in the ZCL [3]. If the *Manufacturer field present* sub-field is set to 0b1, all the following *ClusterID* fields in the *Attribute Record* fields in this command contain a manufacturer-specific cluster corresponding to the *ManufacturerID*.

A.4.2.6.3 Write Attributes command

The Write Attributes command is sent to write attributes of the GPD. The GPD SHALL write all requested attributes that are implemented. If ManufacturerID field is included, all attributes standard ClusterIDs contained in the Write Attributes command SHALL be interpreted as standard attributes defined in the ZCL [3]. They SHALL be written, if implemented, irrespective of this ManufacturerID being supported or not. All attributes of manufacturer-specific ClusterIDs SHALL be interpreted in the context of the ManufacturerID; they are written if ManufacturerID and a particular attribute are implemented.

The command payload for the Write Attributes command is formatted as shown in Figure 149.

Octets	1	0/2	variable	…	0/variable
Data Type	8-bit bitmap	Unsigned 16-bit integer	structure	…	structure
Field name	Options	Manufacturer ID	Write cluster record	…	Write cluster record

Figure 149 – Payload of the GPD Write Attributes command

The Options field is formatted as shown in Figure 144, and the subfields are defined as in A.4.2.6.1.

The *Manufacturer ID* field can take values as defined in [7].

The *Write cluster record* field is formatted as shown in Figure 150.

2	1	variable	Variable	…	variable
Unsigned 16-bit integer	Unsigned 8-bit integer	structure	Structure	…	structure
Cluster ID	Length of record list	Write Attribute record	Write Attribute record	…	Write Attribute record

Figure 150 – Format of the Cluster record field

The *Length of Record List* field indicates the total size in octets of the following Write Attribute record List until the next Cluster ID field. The *Write Attribute Record* field is formatted as shown in Figure 151.

Octet: 2	1	Variable
Unsigned 16-bit integer	8-bit enumeration	variable
AttributeID	Attribute Data Type	Attribute Value

Figure 151 – Format of the Write attribute record field

A.4.2.6.4 Read Attributes command

The command payload for the Read Attributes command is formatted as shown in Figure 143, Figure 144, and Figure 145.

A.4.2.7 Scene commands

On reception of the GPD Recall Scene and GPD Store Scene commands, if supported, the Green Power EndPoint of the sink fills in the *GroupID* parameter of the corresponding ZCL command, before forwarding the command to the application endpoint.

If the sink implements the Translation Table, it SHALL act as follows: if the *GroupID* parameter of the *Zigbee Command payload* field of the Translation Table entry carries the value 0xffff, the *GroupID* for the mapped ZCL command SHALL be derived from the GPD ID, as described in sec. A.3.6.3.3.1. Otherwise, the sink SHALL use the GroupID value provided.

This is also the default recommended behavior for the sinks not implementing the Translation Table.

On reception of a GPD Store Scene command, if supported, the sink SHALL attempt to create a scene. If the Translation Table is supported, the scene SHALL be created for the endpoint(s) as indicated by the *Endpoint* parameter of the Translation Table entry for the triggering GPD Store Scene command, e.g. by sending the corresponding ZCL Store Scene command of the ZCL Scenes cluster. The same endpoint(s) SHALL be added to the GroupID (with the value as explained above), e.g. by sending the ZCL Add group command of the ZCL Groups cluster.

A.4.2.8 Manufacturer-defined GPD commands

The command payload for the manufacturer-defined GPD commands is formatted as shown in Figure 152.

Octets	2	0/Variable
Data Type	16-bit enumeration	Sequence of octets
Field name	Manufacturer ID	Data

Figure 152 – Format of the Manufacturer-defined GPD commands

The *ManufacturerID* field can take values as defined in [7].

The remaining fields are specified per *ManufacturerID* and *CommandID* combination.

If any manufacturer-defined GPD command is implemented by the GPD, it SHALL be indicated in the GPD Commissioning command, if supported, by including the *ManufacturerID* and the supported manufacturer-specific GPD CommandID in the *GPD CommandID list* field; the sub-fields of the *Application information* field SHALL be set accordingly.

A.4.3 GP Devices (GPD)

GP Devices (GPD), i.e. the energy-harvesting devices, have their own device descriptions and identifiers, although many of them have an equivalent in the existing profiles (e.g. GP On/Off Switch is an energy harvesting ZHA or ZBA On/Off Switch).
Dedicated definitions are chosen for GP devices, because they have a different set of mandatory and optional clusters than their normal Zigbee counterparts. Dedicated definitions also allow for additional flexibility in standardizing devices in the future that will only work with energy harvesters.
Furthermore, for efficiency, the limited set of GPD type identifiers (GPD DeviceID) is encoded on 1 octet.
The Master List of Green Power Device description [17] contains the Green Power Device definitions for the *ApplicationID* sub-field of the Extended NWK Frame Control field set to 0b000 or 0b010.
It contains:

- Device name;

- DeviceID;
- Minimal application functionality of the GPD:
    - List of GPD Commands, which are mandatory to be transmitted by this GPD;
      The format of the GPD Commands is defined in the Green Power specification, with the version number as indicated in [17] or later.
    - List of GPD Commands, which are optional to be transmitted by this GPD;
      The format of the GPD Commands is defined in the Green Power specification, with the version number as indicated in [17] or later.
    - For the GP Devices supporting the ZCL functionality
        - And the standard GPD reporting commands 0xA0-xA3 and 0xA6 (see sec. A.4.2.3):
            - List of ZCL clusters, which are mandatory to be supported by this GPD;
              The names of those ZCL clusters are defined in the ZCL [3]; their identifiers are defined in the Master Cluster List [12].
            - List of ZCL cluster attributes, which are mandatory to be supported by this GPD;
              The names, identifier and format of those ZCL cluster attributes are defined in the ZCL [3].
        - And the GPD bidirectional operation commands (see sec. A.4.2.6):
            - List of ZCL cluster attributes, which are mandatory to be readable on this GPD;
              The names, identifier and format of those ZCL cluster attributes are defined in the ZCL [3].
            - List of ZCL cluster attributes, which are mandatory to be writable on this GPD;
              The names, identifier and format of those ZCL cluster attributes are defined in the ZCL [3].
        - [312]And the GPD Compact Attribute Reporting command (0xA8) (see sec. A.4.2.3.6):
            - List of ZCL clusters defined for usage with GPD Compact Attribute Reporting command to-date, with the corresponding cluster attributes, which are mandatory to be reported by a GPD supporting this cluster via GPD Compact Attribute Reporting command and additional attributes mandatory to then be included in the GPD Application Description command carrying Data Point Descriptor for that cluster.
              The names of those ZCL clusters are defined in the ZCL [3]; their identifiers are defined in the Master Cluster List [12]; The names, identifier and format of those ZCL cluster attributes are defined in the ZCL [3].
            - Other clusters and cluster attributes MAY also be supported via the GPD Compact Attribute Reporting command.

In addition to the mandatory ZCL cluster attributes as specified in [13], the GPDs MAY optionally support additional attributes of the same ZCL cluster.

The following rules are specified for the usage of the DeviceIDs defined by the Green Power specification:

- A GPD supporting standard ZCL clusters SHALL only use a GP-defined *DeviceID* != 0xFE, if it supports all the standard ZCL clusters mandatory for this *DeviceID*.
- A GPD supporting only some of the standard ZCL clusters mandatory for a particular *DeviceID* != 0xFE SHALL NOT use that *DeviceID*.

[312] GP multi-sensor v0.9 LB comment #975: https://workspace.zigbee.org/kws/groups/PRO_GP/comments/view_comment?comment_id=975

It SHALL use either: a *DeviceID* whose mandatory ZCL clusters are all supported, or *DeviceID* 0xFE, or a *DeviceID* not mandating any ZCL clusters (e.g. *DeviceID* 0x00 – 0x03) if other requirements for using that *DeviceID* are fulfilled.
It SHALL then follow the rules for listing the supported clusters in the *Application Information*, as defined in sec. A.4.2.1.1.4- A.4.2.1.1.9.

- A GPD supporting standard GPD Data commands is allowed to use GP-defined *DeviceID* != 0xFE, if it supports at least one of the standard GPD Data commands mandatory for this *DeviceID*.
It SHALL then follow the rules for listing the supported GPD commands in the *Application Information*, as defined in sec. A.4.2.1.1.4- A.4.2.1.1.9.

A.4.3.1 GPDs not defined by the Green Power specification

If order to allow for creation of GPD which application functionality is not covered by the current specification, a number of mechanisms are provided.

The application information fields of the GPD Commissioning commands can be used to carry the information about the extended application functionality supported by the GPD, including (additional) standard-defined GPD commands, manufacturer-defined GPD commands, or cluster functionality, standard-defined (see [ZCL]) or manufacturer specific.

A dedicated DeviceID, 0xFE, is reserved for devices with to-date undefined type, which can then announce their application functionality using the mechanisms described in the previous section. However, the GPD Commissioning command extensions can also be used in combination with standard-defined DeviceIDs, to add functionality not mandated by a particular GPD device type.

Note: the cluster-based functionality SHALL only be used for functionality not defined as GPD command.