ZigBee Document 095499

ZigBee PRO Green Power feature Specification

Revision 26

Version 1.0a

May 21st, 2014

Sponsored by: ZigBee Alliance

Accepted for release by:
This document has not yet been accepted for release by the ZigBee Alliance Board of Directors.

Abstract:
This document contains the specification of the Green Power feature.

Keywords:
ZigBee, Green Power, Battery-less, Energy Harvesting, Green Power stub, GreenPower Cluster

Legal Notice

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

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

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

ZigBee Alliance, Inc.
2400 Camino Ramon, Suite 375

San Ramon, CA 94583, USA

Participants

The following is a list of those who were members of the PRO Foundation Working Group leadership when this document was released:

Cam Williams: *Chair*

Jonathan Harros: *Vice-Chair*

Raymond Hicks: *Secretary*

When the document was released, the Green Power Task Group leadership was composed of the following members:

Bozena Erdmann: *Chair*

Kevin Doorakkers: *Vice-Chair*

Bozena Erdmann: *Technical Editor*

Contributions were made to this document by the following members:

Rob Alexander
Peter Burnett
Steven Boeykens
Tony Cave
Nicolas Cochard
Robert Cragie
Kevin Doorakkers
Bozena Erdmann
Chris Gray
Timothy Hirou
Ted Humpal
Ray Jessup
David Kravitz
Tako Lootsma
Nimrod Ilan
Thomas De Prycker
Jonathan Simon
Gilles Thonet
Ludo Tolhuizen
Mads Westerngreen
Bas de Wit
Ross Yu

.

Table of Contents

List of Figures

List of Tables

Revision history

Table 1 shows the revision history for this specification.

Table 1 – Document revision change history

Revision	Version	Description
25	1.0a	Changes since the approved r24 (GP v1.0): • Implemented CCB #1661: Handling of reserved fields in GPD Commissioning command, as resolved in GP v1.0 errata, 12-0624r00; • Updates resulting from the ZigBee Alliance structure change and Green Power TG chairmanship change; • Updated the list of non-certifiable features: TC-LK protection removed; GP Simple generic 1-state switch removed; GP Advanced generic 1-state switch removed.
26	1.0a	Clean version of r25.

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. GreenPower 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 053474r19 (or later release), ZigBee Specification

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

[3] ZigBee document 075123r04, ZigBee Cluster Library Specification

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

[5] ZigBee document 105879, Draft CO2 Level Cluster

[6] ZigBee document 105521r23, Green Power test specification v1.0a

[7] ZigBee document 105850r22, Green Power PICS v1.0a

[8] ZigBee document 053874, ZigBee Manufacturer Code Database

[9] ZigBee document 106138, Recommendation for ZigBee PRO Interoperability Across Profiles

[10] ZigBee document 115337, Green Power SrcID Policy Proposal

[11] ZigBee document 106050r03, ZigBee Device Interworking

[12] ZigBee document 115456r04, Master Cluster List

[13] ZigBee document 120624, Errata for GP 1.0 specification (095499)

[14] ZigBee document 120625, Errata for GP 1.0 Test specification (105521)

[15] ZigBee document 120626, Errata for GP 1.0 PICS (105850)

[16] ZigBee document 120525, Product Details Guidelines

2.1.2 ISO / IEEE Standards Documents

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

[18] 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

[19] ZigBee document 053520, ZigBee Home Automation Profile Specification

[20] ZigBee document 105859, ZigBee Building Automation Profile Specification

[21] ZigBee document 11197, GP best practices for ZHA

[22] 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 Reserved values[1]

To support backwards and forwards compatibility, devices ***should*** ignore any values or bit settings for any reserved field or sub-field. If the field or sub-fields is necessary for interpreting or necessary for use in conjunction with other fields, the whole message can 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.

To enable future growth and ensure backwards and forwards compatibility, any existing devices which encounter any fields applied after the end of a command ***shall*** treat them as reserved fields.

[1] CCB #1661, as resolved in GP v1.0 errata document, 12-0624r00.

3.3 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.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 [17]). 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 [17]).

ZigBee End Device: An IEEE 802.15.4-2003 RFD (Reduced Function Device) or FFD (Full Function Device) (see [17]) 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 GreenPower 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.

(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 without further description for the GP-defined commands, 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 – GPS receiving directly the GPFS in GP frame format sent by GPD, if in the radio range of the GPD.

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 GPPs and GPSs. 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 GP Sinks, which makes the GPS 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.

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

Tunneled mode – GPS receiving the GPFS forwarded by a GPP located in the radio range of the GPD. This forwarding uses a normal ZigBee frame format but a specific ZCL command from the GreenPower cluster: the GP Notification command.

Data GPDF – any GPDF that carries a GPD Command other than GPD Commissioning (0xE0) or GPD Commissioning Reply (0xF0) or GPD Decommissioning (0xE1).

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) or GPD Channel Configuration (0xF3).

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

Green Power Proxy Minimum (GPPm) or **Minimum Proxy** – A GPP that only implements the minimum GP proxy functionality, as defined in section A.3.2.6.

Green Power Sink (**GPS**) or **Sink** – term used for describing any of GP Target or GP Target+ or the Target functionality of the GP Combo (see section A.3.2), referring to the capability to receive and process tunneled GPD commands.

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 GreenPower Cluster, allowing for receiving, processing and executing tunneled GPD commands.

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.

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 Minimum (GPCm) or **Minimum Combo** – A GPC that only implements the minimum GP combo functionality, as defined in section 0.

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).

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
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
GPP	Green Power Proxy device
GPPm	Green Power Proxy Minimum device
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

Table 2 includes a list of GP functionality NOT yet certified.

Table 2 – Not certified GP functionality

Functionality	Reference
Lightweight unicast communication functionality	A.3.2.8
GPD IEEE address functionality	A.3.2.8
GP Simple Generic 2-state Switch	A.4.3
GP Level Control Switch	A.4.3
GP Simple Sensor	A.4.3
GP Advanced Generic 2-state Switch	A.4.3
GP Color Dimmer Switch	A.4.3
GP Light Sensor	A.4.3
GP Occupancy Sensor	A.4.3
GP Door Lock Controller	A.4.3
GP Pressure Sensor	A.4.3
GP Flow Sensor	A.4.3
GP Indoor Environment Sensor	A.4.3

6 Overview

The goal of this specification is to allow for usage of energy-harvesting devices within the ZigBee ecosystem.

Such Green Power Devices, GPD, may harvest different amount 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 Green Power Sink (GPS) which processes and acts upon the information sent by GPD. That GP functionality is GP stub (section A.1) and GreenPower 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.

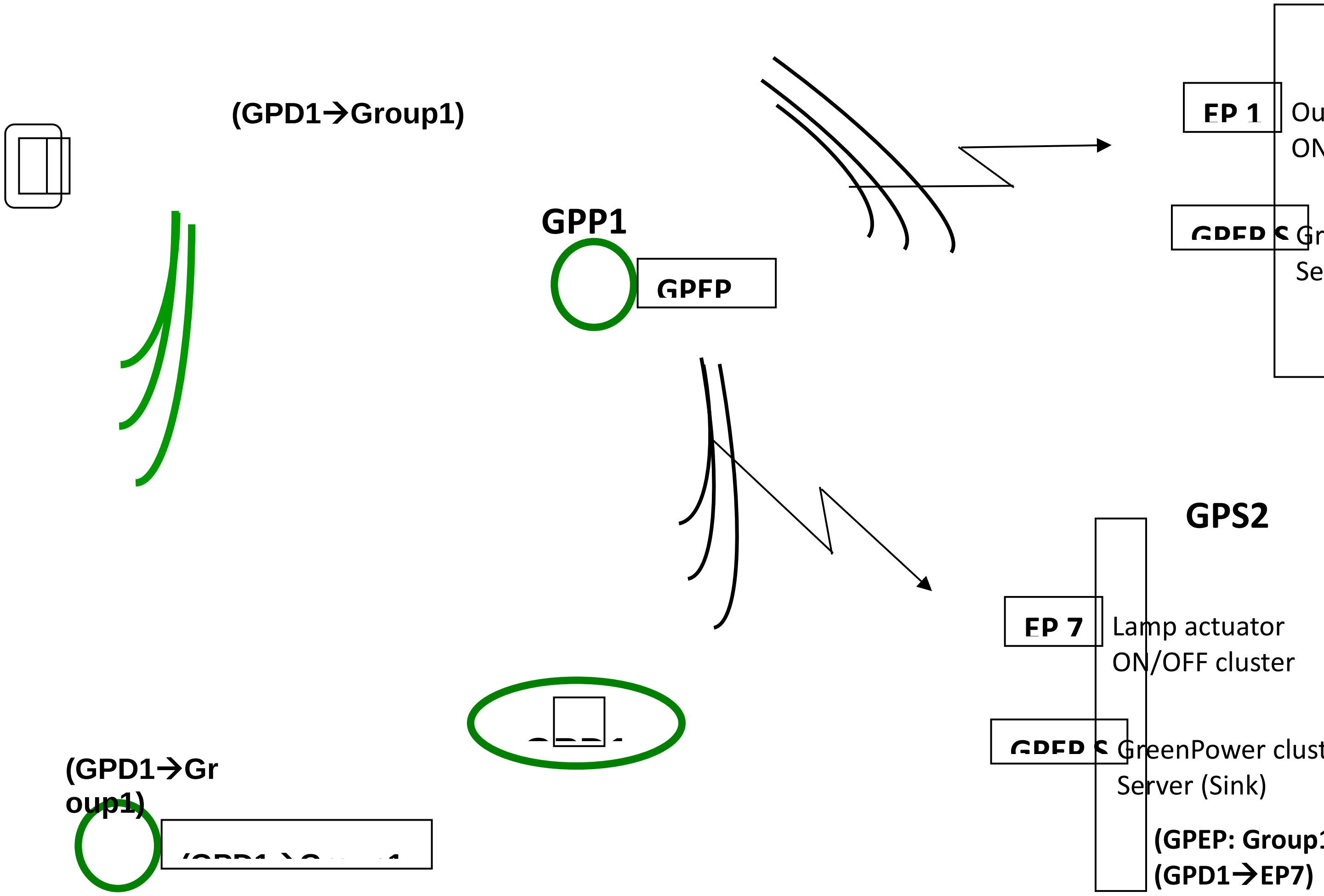


Figure 1 – System overview for the Green Power feature

The Green Power solution relies on the fact, that the future generation of Green Power Sinks (GPSs) to be controlled by the GPD, implements the server side of the GreenPower cluster, to interpret and act upon selected GPD commands. This architectural choice allows for simple operation of the Green Power Proxy (GPP) 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 forwards-compatible with any future GPD types.

The GPSs 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.

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 3 by the Cluster Library Development Board (see also [12]).

Table 3 –Clusters ID allocation for candidate clusters

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

A.1Green 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 GreenPower cluster on GPEP 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 (GPEP) that will use the GP communication mechanism.

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

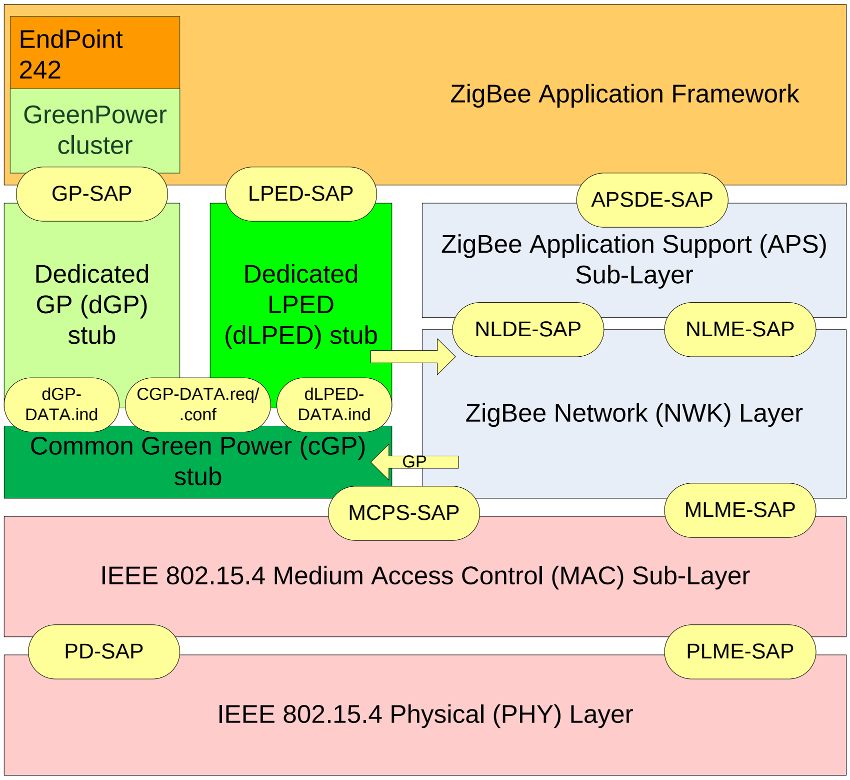

EndPoint 242
GreenPower cluster
ZigBee Application Framework
GP-SAP
LPED-SAP
APSDE-SAP
Dedicated GP (dGP) stub
Dedicated LPED (dLPED) stub
ZigBee Application Support (APS) Sub-Layer
NLDE-SAP
NLME-SAP
dGP-DATA.ind
CGP-DATA.req/ .conf
dLPED-DATA.ind
ZigBee Network (NWK) Layer
Common Green Power (cGP) stub
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 (GPP, GPPm, GPCm, GPC) to send a ZigBee frame with an alias source address and alias sequence number, supplied by the GPEP;
- The ability of GPEP 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 4 – 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	Set of octets	-	The set 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 5 – 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 [17]).
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     {
                        LinkQuality
                        SeqNumber
                        SrcAddrMode
                        SrcPANId
                        SrcAddress
                        DstAddrMode
                        DstPANId
                        DstAddress
                        GP MPDU Length
                        GP MPDU
                        }
```

Table 6 – Parameters of the dGP-DATA.indication

Name	Type	Valid Range	Description
Link quality	Unsigned 8-bit integer	0x00 – 0xff	The link quality 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	Set of octets	-	The set 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 GPEP to request transmission of a GPDF.
- GP-DATA.confirm – provides a mechanism for the GPEP 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 GPEP.
- GP-SEC.request – provides a mechanism for dGP stub to request security data from the GPEP.
- GP-SEC.response – provides a mechanism for the GPEP 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
                            LinkQuality
                            SeqNumber
                            SrcAddrMode
                            SrcPANId
                            SrcAddress
                            ApplicationID
                            GPDFSecurityLevel
                            GPDFKeyType
                            AutoCommissioning
                            RxAfterTx
                            SrcID
                            GPD security frame counter
                            GP CommandID
                            GP ASDU Length
                            GP ASDU
                            MIC
                            }
```

Table 7 – 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
Link quality	Unsigned 8-bit integer	0x00 – 0xff	The link quality 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 – 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.
SrcID	Unsigned 32-bit Integer	0x00000000 – 0xffffffff	The identifier of the GPD entity from which the ASDU was received.
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.
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	Set of octets	-	The set of octets forming the received GPD ASDU.
MIC	Unsigned 16-bit or 32-bit Integer	As specified by the GPDFSecurityLevel parameter	The set of octets forming the MIC for the received GPD MPDU.

A.1.3.1.2 When generated

This primitive is generated and passed to the application in the event of the receipt, by the dGP stub, of a MCPS-DATA.indication primitive from the MAC sub-layer, 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
                    GPD CommandID
                    GPF ASDU Length
                    GPD ASDU
                    GPEP handle
                    gpTxQueue Entry Lifetime
                    }
```

Table 8 – 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 unreserved values as defined in Table 12)
			b5 – 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.
SrcID	Unsigned 32-bit Integer	0x00000000 – 0xffffffff	The identifier of the GPD entity to which the ASDU will be sent if ApplicationID = 0b010.
GPD IEEE address	IEEE address	Any valid	The identifier of the GPD entity to which the ASDU will be sent if ApplicationID = 0b010.
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	Set of octets	-	The set of octets forming the transmitted GPD ASDU.
GPEP handle	Unsigned 8-bit integer	0x00-0xff	The handle used between GPEP and dGP stub, to match the request with the confirmation.
gpTxQueueEntry-Lifetime	Unsigned 16-bit integer	0x0000 – 0xffff	The lifetime of this packet in the gpTxQueue, in ms. For GPD Commissioning Reply, initialized to *CommissioningWindow*. 0x0000 indicates immediate transmission. 0xffff indicates infinity.

A.1.3.2.2 When generated

This primitive is generated by the GPEP and passed to the dGP stub when a GPDF is to be sent to the GPD identified by the GPD SrcID/GPD IEEE address.

A.1.3.2.3 Effect on receipt

Upon receipt of this primitive the dGP stub shall add the GPDF to the gpTxQueue.

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 9 – 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 GPEP 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 GPEP 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 GPEP 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
                    GPDFSecurityLevel
                    GPDFKeyType
                    GPDSecurityFrameCounter
                    dGP stub handle
                    }
```

Table 10 – 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.
SrcID	Unsigned 32-bit Integer	0x00000001 – 0xfffffffe	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.
GPDFSecurityLevel	8-bit enumeration	0x01 – 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 GPEP on reception of protected GPDF.

A.1.3.4.3 Effect on receipt

Upon receipt of this primitive the GPEP is informed about reception of protected GPDF. The GPEP responds with GP-SEC.response primitive, with appropriate status, based on the GPEP client/server functionality, the operational/commissioning mode the GPEP is in and the content of Proxy/Sink Table and Security 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
                    GPDFSecurityLevel
                    GPDFKeyType
                    GPDKey
                    GPDSecurityFrameCounter
                    gppSecurityWindow
                    }
```

Table 11 – Parameters of the GP-SEC.response

Name	Type	Valid Range	Description
Status	8-bit enumeration	Any valid	The status code, as returned by the GPEP. The following are supported: MATCH DROP_FRAME PASS_UNPROCESSED
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.
SrcID	Unsigned 32-bit Integer	0x00000001 – 0xfffffffe	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.
GPDFSecurityLevel	8-bit enumeration	0x01 – 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 8-bit or 32-bit Integer	As specified by the *GPDFSecurityLevel* parameter	The security frame counter value to be used for GPDF security processing.
gppSecurityWindow	Unsigned 8-bit integer	0x00-0xff	The *gppSecurityWindow* value to be used by the GP stub for security processing of this incoming frame.

A.1.3.5.2 When generated

This primitive is generated by the GPEP 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, 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 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.

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 GreenPower cluster

Please note, that the GreenPower 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.

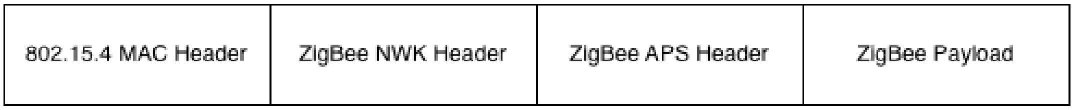
The supplied *UseAlias*, if set to 0b1, indicates that the supplied *SrcAddr* and *NWKSeqNumb* parameters shall be used; otherwise they can be ignored.

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. The NIB *nwkSequenceNumber* ***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].



802.15.4 MAC Header	ZigBee NWK Header	ZigBee APS Header	ZigBee Payload

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	0/4
Frame Control	Sequence Number	Addressing fields	NWK Frame Control	Extended NWK Frame Control	GPD SrcID	Security frame counter
802.15.4 MAC Header			GP stub NWK Header			

Figure 4 – GPDF Frame Format (part 1)

Variable	0/2/4	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

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.

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 12.

Table 12 – Values of *Frame Type* used in combination with *ZigBee Protocol Version* = 0x3

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

If the *FrameType* 0b01 (Maintenance frame) is used, then the *GPD SrcID* field and the security fields (*Security frame counter* and *MIC*) ***shall not*** be present. If the GPDF is sent from the GPD, the *Extended NWK Frame Control* field ***should*** be omitted. If the GPDF is sent to the GPD, the *Extended NWK Frame Control* field ***may*** be omitted. In both cases, the *NWK Frame Control Extension* sub-field ***shall*** be set accordingly.

If the *FrameType* 0b00 is used, the GPDF ***shall*** be formatted as follows.

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.

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 command structure. The current specification defined the GPDF command structure 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.

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 (GP), 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.

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.

If *ApplicationID* is set to 0b000 and 0b010, the *Security Level* sub-field can have values as defined in Table 13. 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 0b01, the *Security Frame counter* field is not present, the MAC *sequence number* field carries the 1LSB of the frame counter, and the *MIC* field is present, has the length of 2B, and carries the 2LSB of the Message Integrity Code (see sec. A.1.5.4.3). 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.1.5.4.3). 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.1.5.4.3).

The *SecurityKey* sub-field indicates the type of the key used for frame protection by this GPD. The *Security Key* sub-field, if set to 0b1, indicates an individual key (*KeyType* 0b100 or 0b111). If set to 0b0, it indicates a shared key (*KeyType* 0b011, 0b010 or 0b001) or no key.

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 after *gpdRxOffset*, for a device-specific duration, but not shorter than *gpdMinRxWindow*. If the value of this sub-field is 0b0, then the GPD will not enter the receive mode after sending this particular GPDF frame. Default value to be used on reception, if the *Extended NWK Frame Control* field is not present, is 0b0.

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 GPP. 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 *FrameType* sub-field is set to 0b00 and the *ApplicationID* sub-field of the *Extended NWK Frame Control* field is set to 0b000 (or not present). It is also present if the *FrameType* sub-field is set to 0b01, the *NWK Frame control Extension* sub-field is set to 0b1, and the *ApplicationID* sub-field of the *Extended NWK Frame Control* field is set to 0b000.
The *GPDSrcID* field carries the unique identifier of the GPD, to/by which this GPDF is sent.

The value of 0x00000000 indicates unspecified. The value of 0xffffffff indicates all. The values 0xfffffff9 – 0xfffffffe are reserved.

The *GPDSrcID* field is not present if the *FrameType* sub-field is set to 0b01 and the Extended *NWK Frame control* sub-field is set to 0b0. Unique identification of the GPD by an address is not required then.
The *GPDSrcID* field is not present if the *ApplicationID* sub-field of the *Extended NWK Frame Control* field is set to 0b010. The GPD is then identified by its IEEE address, which is then carried in the corresponding MAC address field, source or destination for the GPDF sent by or to the GPD, respectively.
The *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 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.1.4.1.6 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 Comman-dID	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 GreenPower cluster (see Table 48 and Table 49). The *GPD command payload* field is of type set of octets, and its presence and length is defined by the value of the *GPD CommandID* field.

A.1.4.1.7 MIC field

The *MIC* field carries the Message Integrity Code for this message, calculated as specified in sec. A.1.5.4.3. Its presence and length is dependent on the value of *ApplicationID* and *SecurityLevel* (see A.1.4.1.3).

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.

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

On receipt of the GP-DATA.request primitive, the dGP stub shall check the *gpTxQueue*. If the *gpTxQueue* already has an entry for the GPD ID (i.e. GPD SrcID/GPD IEEE address) in the GP-DATA.request, the previous GPDF is overwritten and GP-DATA.confirmation with the Status ENTRY_REPLACED is provided to the GPEP. If the *gpTxQueue* has no previous entries for this GPD SrcID/GPD IEEE address and it has empty entries, the GPDF is added to the *gpTxQueue* and GP-DATA.confirmation with the Status ENTRY_ADDED is provided to the GPEP. If the *gpTxQueue* has no previous 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

In gpTxQueue, GPDF are stored for transmission to GPD.

In its gpTxQueue, each GPP shall have a maximum of only one pending GPDF frame per GPD ID.

Each entry in the gpTxQueue entry shall have a gpTxQueueEntryLifetime parameter associated, initiated by the value in the GP-DATA.request. When this timeout elapses, the GP-DATA.confirm with the Status ENTRY_EXPIRED, the entry is cleared and can be used for any GPDF for any GPD ID.

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 a GPDF in response to a GPDF with *RxAfterTx* sub-field set, if any present in the gpTxQueue for this GPD ID. It is measured from the start of the reception of the first GPDF in a given GPFS.

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

A.1.5.2.1.3 gpTxDuration

The *gpTxDuration* is the maximum allowed transmission time for the GP stub after *gpTxOffset*. Thus, depending on the GPDF length, the GP stub may send the GPDF more than once, to increase the reliability of communication. It is measured from the start of the transmission of the first 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*** check the *SecurityLevel*. If the *SecurityLevel* is not supported, the dGP stub ***shall*** silently drop the frame. If *SecurityLevel* is supported and has the value of 0b00-0b10, 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. On receipt of GP-SEC.response with Status MATCH, the GP stub security-processes the received GPDF, as described in A.1.5.4.4.

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.

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, 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 for this GPD ID. 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 to commence after *gpTxOffset*, by sending CGP-DATA.request, with *UseCSMA* parameter set to FALSE. On reception of the MCPS-DATA.confirmation, the dGP calls GP-DATA.confirmation with Status value copied from the MCPS-DATA.confirmation.

Subsequently, 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 0b01 – 0b11, the dGP calls GP-DATA.indication with the Status SECURITY_SUCCESS.

A.1.5.3 Security parameters

The dGP stub of a GPP ***shall*** support all security levels defined in the GP specification.

The dGP stub of a GPS ***shall*** support all security levels above and including the application- and product-specific minimum security level, as indicated in the *gpsSecurityLevel* attribute.

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 gpSecurityLevel

The gpSecurityLevel can take the values as defined in Table 13.

Table 13 – Values of gpSecurityLevel

Value	Description
0b00	No security
0b01	1LSB of frame counter and short (2B) MIC only
0b10	Full (4B) frame counter and full (4B) MIC only
0b11	Encryption & full (4B) frame counter and full (4B) MIC

A.1.5.3.3 gpSecurityKeyType

The gpSecurityKeyType can take the values as defined in Table 14.

Table 14 – 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.1.5.3.3.1. 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 A.1.5.3.3.1, 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.
0b101-0b110	Reserved		

Value	Description	Comment	Security properties
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 A.1.5.3.3.2. 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 A.1.5.3.3.2, 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.1.5.3.3.1 GPD group key (0b011) derivation

The HMAC keyed hash function, as defined in [18], is used to derive the GPD group key (0b011).

K_{GP}= HMAC(K, 'GP')$_{16}$

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 GPS; the proxies receive the correct key in the GP Pairing command.

A.1.5.3.3.2 Individual GPD key derivation

The HMAC keyed hash function, as defined in [18], 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);
  - 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 GPS; the proxies receive the correct key in the GP Pairing command.

A.1.5.3.3.3 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;

      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;
    - *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*).
    - *Security control* field taking the value as described in sec. A.1.5.4.1.

A.1.5.3.3.4 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.1.5.3.4 gppSecurityWindow

Number of times the GP stub of the GPP receiving a GPDF secured with *SecurityLevel* 0b01 is allowed to increment the upper part of the GPD security frame counter upon security processing failure on the first try.

The value is passed into the GP stub as *gppSecurityWindow* parameter being part of the GP-SEC.response primitive.

The default value is 0x00.

A.1.5.4 Security operation of the GP stub

A.1.5.4.1 Constructing AES Nonce

The AES nonce, defined by the ZigBee specification 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.

Frame counter parameter shall take the value:

- for the incoming secured GPDF: 4B frame counter for this GPD, part or whole of which is being transmitted in the GPDF:
  - if SecurityLevel was 0b01: the frame counter value is derived as described in A.3.7.2.4
- 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.1.5.4.2 Initialization

If the *SecurityLevel* field of the GPDF has the value 0b01, the following transformation applies.

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

Payload = GPD CommandID || GPD Command Payload.

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

Header = MAC sequence number || MAC addressing fields || NWK Frame Control || Extended NWK Frame Control || SrcID.

whereby

- for the MAC sequence number field as part of the *Header*
  - ▪ In case of an incoming frame, the MAC sequence number from the received frame is used.
  - ▪ In case of an outgoing frame, 1LSB of the Security Frame Counter is used for security processing.
    Note: the 1LSB of the Security Frame Counter is independent of the *macDSN* attribute the MAC layer will use to transmit the frame.
- MAC addressing fields = are as in the received frame / as requested by the application;
- SrcID field = as in the received frame / as requested by the application (i.e. only for ApplicationID = 0b000).

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.

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

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

whereby the SrcID field is only present if the *ApplicationID* = 0b000.

A.1.5.4.3 Outgoing frames encryption and authentication

Determine the security level, as described in A.1.5.2.2, and perform initialization, as described in A.1.5.4.2.

A.1.5.4.3.1 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.1.5.4.1.
- 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 inA.1.5.4.2,
  - and the octet string *m* shall be the string *Payload*, as defined in A.1.5.4.2,
- Otherwise if the security level, as indicated by the *gpdSecurityLevel* parameter equal to 0b10 or 0b01, does not require encryption,
  - the octet string *a* shall be the string *Header* || *Payload*, as defined in A.1.5.4.2,
  - 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.1.5.4.3.2 Constructing protected GPDF

For transmission of the protected GPDF:

- If the security level, as indicated by *gpdSecurityLevel* = 0b01:
  - The fields GPD *CommandID* and *GPD Command Payload* remain unmodified;
  - 2 LSB of *U* are inserted into GPDF *MIC* field.
  - Then, the data unit is passed down using the CGP-DATA.request.
    The MAC layer will fill the MAC *Sequence Number* field with the value of the *macDSN* attribute of the MAC PIB.
    Note: the macDSN attribute is independent of the 1LSB of the security frame counter used to protect the frame.
- 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.

A.1.5.4.4 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.1.5.4.2.

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.1.5.4.1.
- 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* 0b01 - 0b11), calculate the *U*, as defined in A.1.5.4.3.1, taking the decrypted *GPD CommandID* and *GPD Command Payload* fields as *Payload*, and the *Header* fields as defined in A.1.5.4.2. 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.1.5.4.4.

A.1.5.4.4.1 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

- or *SecurityLevel* = 0b01 and *gppSecurityWindow* = 0,

dGP stub calls GP-DATA.indication with Status AUTH_FAILED and carrying the protected GPD CommandID and GPD Command Payload.

Otherwise, if the authentication is not successful and SecurityLevel=0b01 and if *gppSecurityWindow* parameter >0, the *gppSecurityWindow* is decremented and Frame Counter is modified as follows: the second LSB of the Frame Counter used in the previous run is incremented by 1, and the LSB is over-written with the MAC sequence number field from the received GPDF. Then, the processing as described in A.1.5.4.4 is performed.

A.1.5.5 Security test vectors for ApplicationID = 0b000 and a shared key

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

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
- 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 =0b10] → [0b10001110] 0x8e
  - GPD SrcID = 0x87654321
  - Security Frame Counter = 0x00000002
- Application fields:
  - GPD CommandID = 0x20 (OFF)
  - No data payload

A.1.5.5.2 SecurityLevel=0b01

A.1.5.5.2.1 Transmitted packet

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

Transmitted packet
12 01 08 02 FF FF FF FF 8E 08 21 43 65 87 20 B9 B3

Note: even for SecurityLevel = 0b01, 4B MIC (*U*) is calculated, of which only part is transmitted in the packet.

A.1.5.5.2.2 Inputs

- NWK fields:
  - Extended NWK FC = [Direction = 0b0 || RxAfterTx = 0b0 || SecurityKey = 0b0 ||SecurityLevel = 0b01 || ApplicationID = 0b000] →0b00001000 → 0x08

A.1.5.5.2.3 GP Security Calculation

Definitions

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

a = header || Payload

Header = MAC sequence number || MAC addressing fields || NWK FC || NWK_EXT FC || SrcID.
header = 0x02 || 0xffff || 0xffff || 0x8e || 0x08 || 0x87654321
header = [0x02, 0xff, 0xff, 0xff, 0xff, 0x8e, 0x08, 0x21, 0x43, 0x65, 0x87]

payload = 0x20

a = 0x02 || 0xffff || 0xffff || 0x8e || 0x08 || 0x87654321 || 0x20
a = [0x02, 0xff, 0xff, 0xff, 0xff, 0x8e, 0x08, 0x21, 0x43, 0x65, 0x87, 0x20]

Calculation
l(a) = 0x0c
L(a) = 0x00 0x0c

AddAuthData = L(a) || a || padding
AddAuthData = [0x00, 0x0c, 0x02, 0xff, 0xff, 0xff, 0xff, 0x8e, 0x08, 0x21, 0x43, 0x65, 0x87, 0x20, 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 = **D6A4B3B9**
MIC = 2LSB of U = 0xB3B9 = [0xB9, 0xB3]

A.1.5.5.3 SecurityLevel=0b10

A.1.5.5.3.1 Transmitted packet

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

Transmitted packet
18 01 08 02 FF FF FF FF 8E 10 21 43 65 87 02 00 00 00 20 0F C0 B0 79

A.1.5.5.3.2 Inputs

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

A.1.5.5.3.3 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 = 0x8e || 0x10 || 0x87654321 || 0x00000002
header = [0x8e, 0x10, 0x21, 0x43, 0x65, 0x87, 0x02, 0x00, 0x00, 0x00]

payload = 0x20

a = 0x8e || 0x10 || 0x87654321 || 0x00000002 || 0x20
a = [0x8e, 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, 0x8e, 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 = **0x79B0C00F**
MIC = FULL U = 0x79B0C00F= [0x0F, 0xC0, 0xB0, 0x79]

A.1.5.5.4 SecurityLevel=0b11

A.1.5.5.4.1 Transmitted packet

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

Transmitted packet
18 01 08 02 FF FF FF FF 8E 18 21 43 65 87 02 00 00 00 83 0F 98 8F C2

A.1.5.5.4.2 Inputs

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

A.1.5.5.4.3 GP Security Calculation

Definitions
- 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 = 0x8e || 0x18 || 0x87654321 || 0x00000002
header = [0x8e, 0x18, 0x21, 0x43, 0x65, 0x87, 0x02, 0x00, 0x00, 0x00]

payload = 0x20

a = 0x8e || 0x18 || 0x87654321 || 0x00000002
a = [0x8e, 0x18, 0x21, 0x43, 0x65, 0x87, 0x02, 0x00, 0x00, 0x00]

m = 0x20

Calculation
l(a) = 0x0a
L(a) = 0x00 0x0a

AddAuthData = L(a) || a || padding
AddAuthData = [0x00, 0x0a, 0x8e, 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, 0x8e, 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, 0x8e, 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 = **0xB64AECD2**
MIC = FULL U = 0xB64AECD2 = [0xD2, 0xEC, 0x4A, 0xB6]

Cipher = **0x83**

A.1.5.6 Security test vectors for ApplicationID = 0b000 and an individual key

A.1.5.6.1 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 =0b10] → [0b10001110] 0x8e
  - GPD SrcID = 0x87654321
  - Security Frame Counter = 0x00000002
- Application fields:
  - GPD CommandID = 0x20 (OFF)
  - No data payload

A.1.5.6.2 SecurityLevel=0b01

Extended NWK FC = [Direction = 0b0 || RxAfterTx = 0b0 || SecurityKey = 0b1 || SecurityLevel = 0b01 || ApplID = 0b000] →0x28

Over the air packet**:**

12 01 08 02 FF FF FF FF 8E 28 21 43 65 87 20 4F 08

Note: even for SecurityLevel = 0b01, 4B MIC (*U*) is calculated, of which only part is transmitted in the packet.

A.1.5.6.3 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 8E 30 21 43 65 87 02 00 00 00 20 E5 B4 5B 52

A.1.5.6.4 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 8E 38 21 43 65 87 02 00 00 00 83 09 48 03 91

A.1.5.7 Security test vectors for ApplicationID = 0b000 and bidirectional operation

A.1.5.7.1 Common settings

For all frames

- NWK Frame Type sub-field = 0b00
- ZigBee Protocol Version sub-field = 0b0011
- Auto-commissioning sub-field = 0b0
- Extended NWK Frame Control Present 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 }

<u>For incoming frames (from GPD to GPPD / GPS)</u>

- RxAfterTx sub-field = 0b1
- Direction sub-field = 0b0
- MAC Seq Nbr
  - For SecurityLevel = 0b10 or 0b11: 0x01
  - For SecurityLevel = 0b01: 0x11 being LSB of Security Frame Counter
- GPD CommandID = 0x20 (OFF)
- GPD Command payload = ∅ (No payload)

<u>For outgoing frames (from GPP/GPS to GPD)</u>

- RxAfterTx sub-field = 0b0
- Direction sub-field = 0b1
- MAC Seq Nbr = 39
  Note: for SecurityLevel = 0b01: 0x11 (LSB of Security Frame Counter) is used for MIC calculation!
- GPD CommandID = 0xF3 (Channel Configuration)
- GPD Command payload = 0x00 (channel 11)

A.1.5.7.2 Security test vectors for a shared key

<u>For all test vectors with a shared security key :</u>

- Security Key sub-field of Extended NWK Frame Control field = 0b0 (shared key)

A.1.5.7.2.1 SecurityLevel = 0b01

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

0x12 0x01 0x08 0x11 0xFF 0xFF 0xFF 0xFF 0x8C 0x48 0x21 0x43 0x65 0x87 0x20 **0x16 0x0B**

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

0x13 0x01 0x08 0x11 0xFF 0xFF 0xFF 0xFF 0x8C 0x88 0x21 0x43 0x65 0x87 0xF3 0x00 **0x6C 0xFD**

Full 4B MIC: 0x4782**FD6C**

A.1.5.7.2.2 SecurityLevel = 0b10

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

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**

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

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.1.5.7.2.3 SecurityLevel = 0b11

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

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

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

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

Outgoing frame (GPP/GPS to GPD)

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

Encrypted data: 0x9E 0x7E

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

A.1.5.7.3 Security test vectors for an individual key

For all test vectors with an individual key:

- Security Key sub-field in NWK Ext field = 0b1 (individual key)

A.1.5.7.3.1 SecurityLevel = 0b01

Incoming frame (GPD to GPP / GPS)

0x12 0x01 0x08 0x11 0xFF 0xFF 0xFF 0xFF 0x8C 0x68 0x21 0x43 0x65 0x87 0x20 **0x43 0x82**

Outgoing frame (GPP/GPS to GPD)

0x13 0x01 0x08 0x11 0xFF 0xFF 0xFF 0xFF 0x8C 0xA8 0x21 0x43 0x65 0x87 0xF3 0x00 **0x71 0x15**

Full 4B MIC: **0xFA601571**

A.1.5.7.3.2 SecurityLevel = 0b10

Incoming frame (GPD to GPP / GPS)

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**

Outgoing frame (GPP/GPS to GPD)

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.1.5.7.3.3 SecurityLevel = 0b11

Incoming frame (GPD to GPP / GPS)

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

Encrypted data: 0x2A

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

Outgoing frame (GPP/GPS to GPD)

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: 0x9E 0x7E

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

A.1.5.8 Security test vectors for key derivation

A.1.5.8.1 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.1.5.8.2 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.1.5.9 Security test vectors for TC-LK protection

A.1.5.9.1 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}

GPD key MIC = {0x61 0xF1 0x63 0xA9}

A.1.5.9.2 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}

GPD key MIC = {0x67 0x31 0x42 0x6A}

A.1.5.9.3 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}

GPD key MIC = {0x27 0x55 0x9F 0x75}

A.1.5.10 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.

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 [10].

The following SrcID values are reserved: 0x00000000 (used for none/undefined), 0xffffffff (used for all/any), and all in the range 0xfffffff9-0xfffffffe (reserved).

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 GPP perform filtering and tunneling based solely on the SrcID.

If GPD supports *ApplicationID* = 0b010, the GPD is identified by its IEEE address.

A.1.6.3 GPD bidirectional operation

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

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 5 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.
It has the value of 0.576 ms that corresponds to the Channel Configuration GPDF of 18B.

GPD vendors shall implement reception window duration that corresponds to the actual GPD frame size to be received by this GPD, which may never be shorter than *gpdMinRxWindow*.

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

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 13.

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

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 14.

A.1.6.4.3 gpdSecurityKey

The security key itself.

Note: if the GPD device comes with a default 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.1.5.4.1).

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.

For gpdSecurityLevel 0b01, the MAC sequence number field **shall** carry the 1LSB of the gpdSecurityFrameCounter.

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.1.5.4.1- A.1.5.4.3 and A.1.5.5.

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 = 0b01
  - The SecurityLevel and SecurityKeyType value in the received frame shall be as for the triggering frame;
  - For security processing, the MAC sequence number field of the received frame shall be replaced with 1LSB of the last security frame counter value used; the same frame counter value shall be used as part of the nonce.
  - The security processing shall be successful.
- if gpdSecurityLevel = 0b01 – 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) are recommended to support incremental sequence numbers.

Only the GPDs that do not have enough energy for storing the sequence number in NVM over the periods when no energy is available ***may*** use random sequence numbers.

For GPDs that support security (gpdSecurityLevel > 0b00), 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 hundred 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.).

A.1.7.2.1 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 that 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 *gpdRxOffset* (see sec. A.1.6.3.1).

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.

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 *GPD Security key request* is set to 0b1. The GPD ***should*** only request the key by setting *GPD Security key request* to 0b1, if it supports security, i.e. if the *Security level capabilities* sub-field of the *Extended Options* field of the GPD Commissioning command is set to 0b01 – 0b11.

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

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 Rx-capable, 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.

If the GPD is capable of exchanging the security key encrypted, it ***shall*** set the GPD Key encryption sub-field of the Extended Options field of the GPD Commissioning command to 0b1, if at least one of the sub-fields *GP Security Key request* or *GPD key present* 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.

A.2ZigBee 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 GreenPower 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 DeviceIDs (analogous to the profile-agnostic Range extender, DeviceID = 0x0008) for the following GP infrastructure device types as defined in Table 15.

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:

[23] 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 [23]

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 [23], 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 [23]

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 GPEP

A.2.4.1 Modify the "Device application" definition in section 1.4.1.2, p. 9, of [23]

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 [23]

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 [23]

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 [23]

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 [23]

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 [23]

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

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 sequence number to be used for this NSDU. If the *UseAlias* parameter has a value of FALSE, the *AliasSeqNumb* parameter is ignored.

A.2.5.3.3 Modify section 2.2.4.1.1.3 Effect on Receipt, p. 25ff, of [23], 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.

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

If the TxOptions parameter specifies that secured transmission is required, the APS sub-layer shall use the security service provider (see 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 [23]

A.2.5.4.1 Modify section 3.2.1.1.1, p. 264, of [23]

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 [23]

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 [23]

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 [23]

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 [23]

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.3GreenPower cluster

A.3.1 Overview

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

A.3.2 GP infrastructure devices

GP infrastructure devices are the devices receiving and forwarding the frames from and/or controlled by the frames from the GPDs. The Device IDs used by GP specification are defined in [11] and listed in Table 15.

The GreenPower cluster ***shall*** use ClusterID 0x0021.

The GreenPower 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 GreenPower cluster messages.

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.

As described in the definitions section, the joint term “GP Sink (GPS)” is used when the exact GPT/GPT+/GPC/GPCm capability is not of importance (see sec. 3.4).

Table 15 – List of GP infrastructure devices

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

A.3.2.1 GP Target device

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

Table 16 – Functionality of GP Target device

Server side (if supported by device)	Client side
Mandatory	
Selected GreenPower cluster commands (see Table 25) and GP functionality (see Table 24)	Selected GreenPower cluster commands (see Table 25) and GP functionality (see Table 24)
Optional	

The GP Target DeviceID (see Table 15) implements the server side of the GreenPower cluster on the reserved end point GPEP (see sec. A.3.6.1) with the selected commands of the client side of the GreenPower cluster (see Table 25), 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 GPS's own applications;
- Ability to (de-)register at GPPs (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).

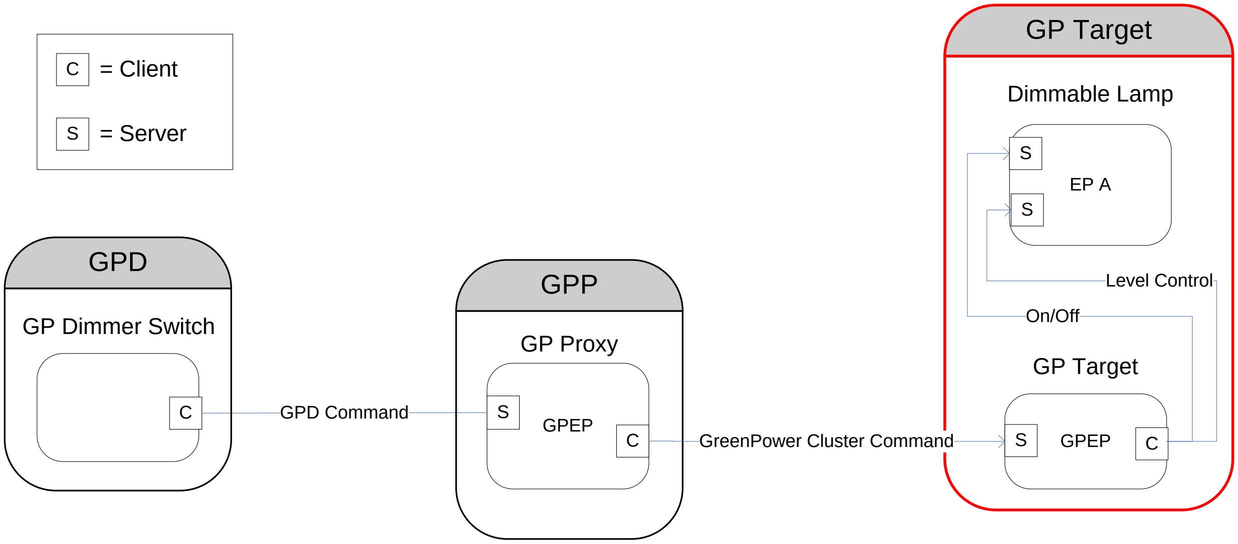


Figure 13 – Example of GP Target device usage

A.3.2.2 GP Target+ device

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

Table 17 – Functionality of GP Target+ device

Server side	**Client side**
Mandatory	
Selected GreenPower cluster commands (see Table 25) and GP functionality (see Table 24)	Selected GreenPower cluster commands (see Table 25) and GP functionality (see Table 24)
	Rx GP stub
Optional	
Tx GP stub	

A GP Target+ DeviceID (see Table 15) requires implementation of both the server side of the GreenPower cluster on the reserved end point GPEP (see sec. A.3.6.1) with the selected commands of the client side of the GreenPower cluster (see Table 25), 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 GreenPower 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.

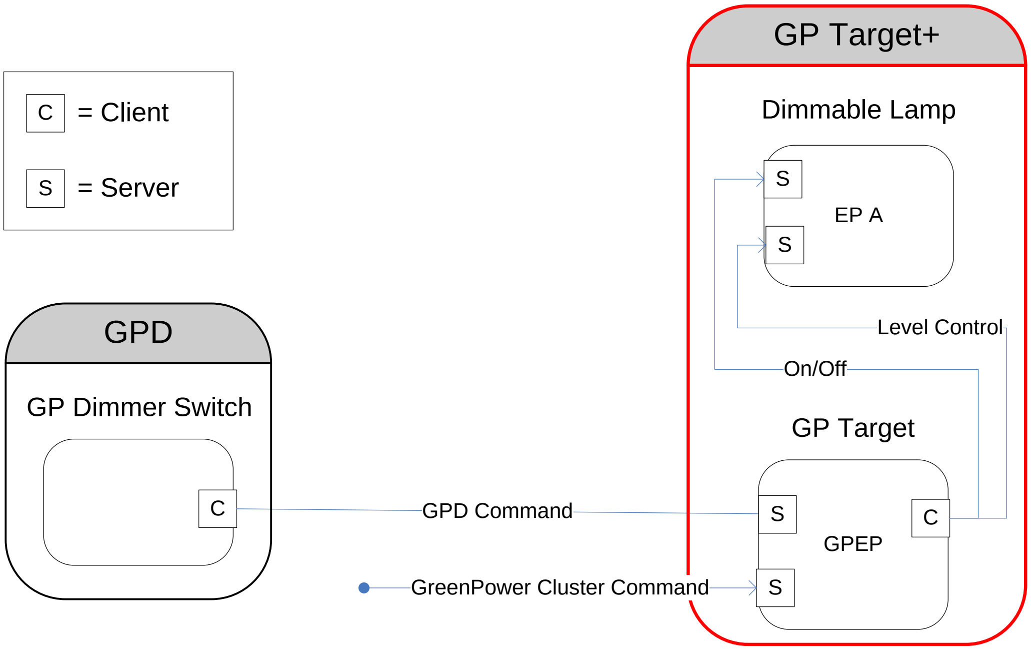


Figure 14 – Example of GP Target+ device usage

A.3.2.3 GP Proxy device

The functionality supported by the GP Proxy device is defined in Table 18.

Table 18 – Functionality of GP Proxy device

Server side	Client side
Mandatory	
Selected GreenPower cluster commands (see Table 25) and GP functionality (see Table 24)	Selected GreenPower cluster commands (see Table 25) and GP functionality (see Table 24)
Tx GP stub	Rx GP stub
Optional	

A GPP is a normal ZigBee device, in most cases a ZR, which implements on its reserved end point GPEP (see sec. A.3.6.1) the GP Proxy DeviceID (see Table 15) with the selected commands of the GreenPower cluster (see Table 25), and a GP stub. GPP has the following GP proxy capabilities:

- Ability to receive any GP frame in direct mode when the GPP 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 GPS devices a GP Notification command with the received GP frame;
- Ability to receive acknowledgements from the check if the GPS 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 GPS 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.

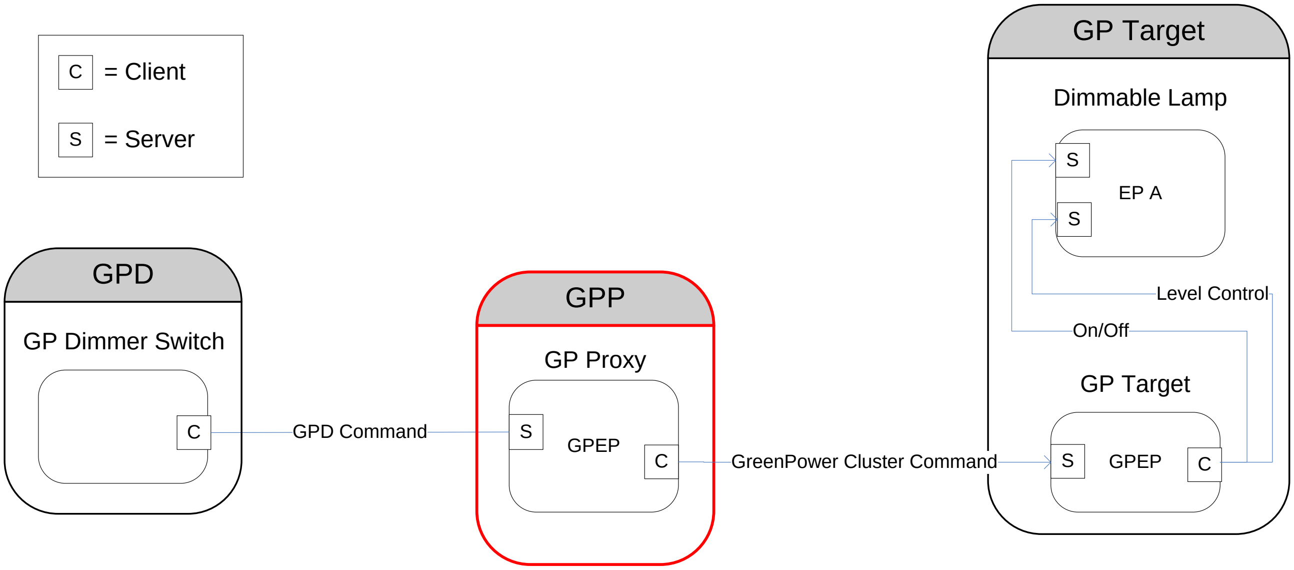


Figure 15 – Example of GP Proxy device usage

A.3.2.4 GP Combo device

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

Table 19 – Functionality of GP Combo device

Server side	Client side
Mandatory	
Selected GreenPower cluster (see Table 25) and GP functionality (see Table 24)	Selected GreenPower cluster commands (see Table 25) and GP functionality (see Table 24)
Tx GP stub	Rx GP stub
Optional	

A GPP can also be at the same time a GPS device. In this case the GPP implements the GP Combo DeviceID (see Table 15) on the GPEP (see sec. A.3.6.1) with selected commands of the GreenPower cluster (see Table 25), and the GP stub. It has all the capabilities of both GPT+ and GPP, 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 GreenPower 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.

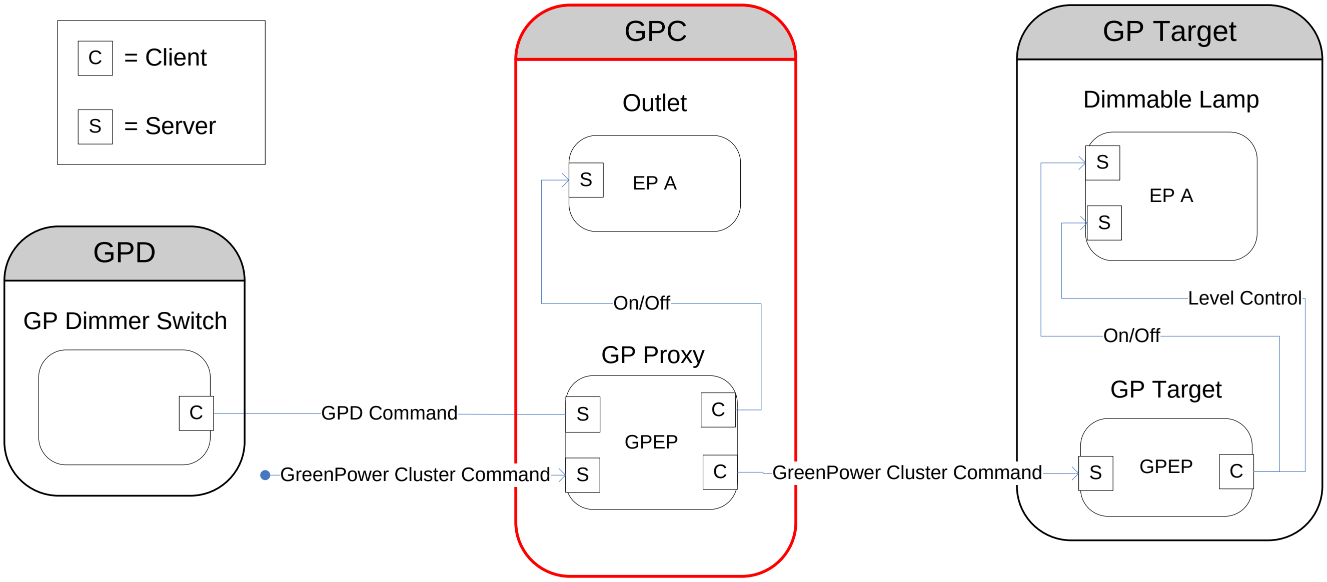


Figure 16 – Example of GP Combo device usage

A.3.2.5 GP Commissioning Tool

The functionality supported by the GP Commissioning Tool device is defined in Table 20.

Table 20 – Functionality of GP Commissioning Tool device

Server side	Client side
Mandatory	
Selected GreenPower cluster commands	
Tx GP stub, Rx GP stub	
Optional	

A GPCT is a regular ZigBee device, in most cases a ZR, which implements on its reserved end point GPEP (see sec. A.3.6.1) the GP Commissioning Tool DeviceID (see Table 15).

GPCT has the following GP proxy 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 GreenPower 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.

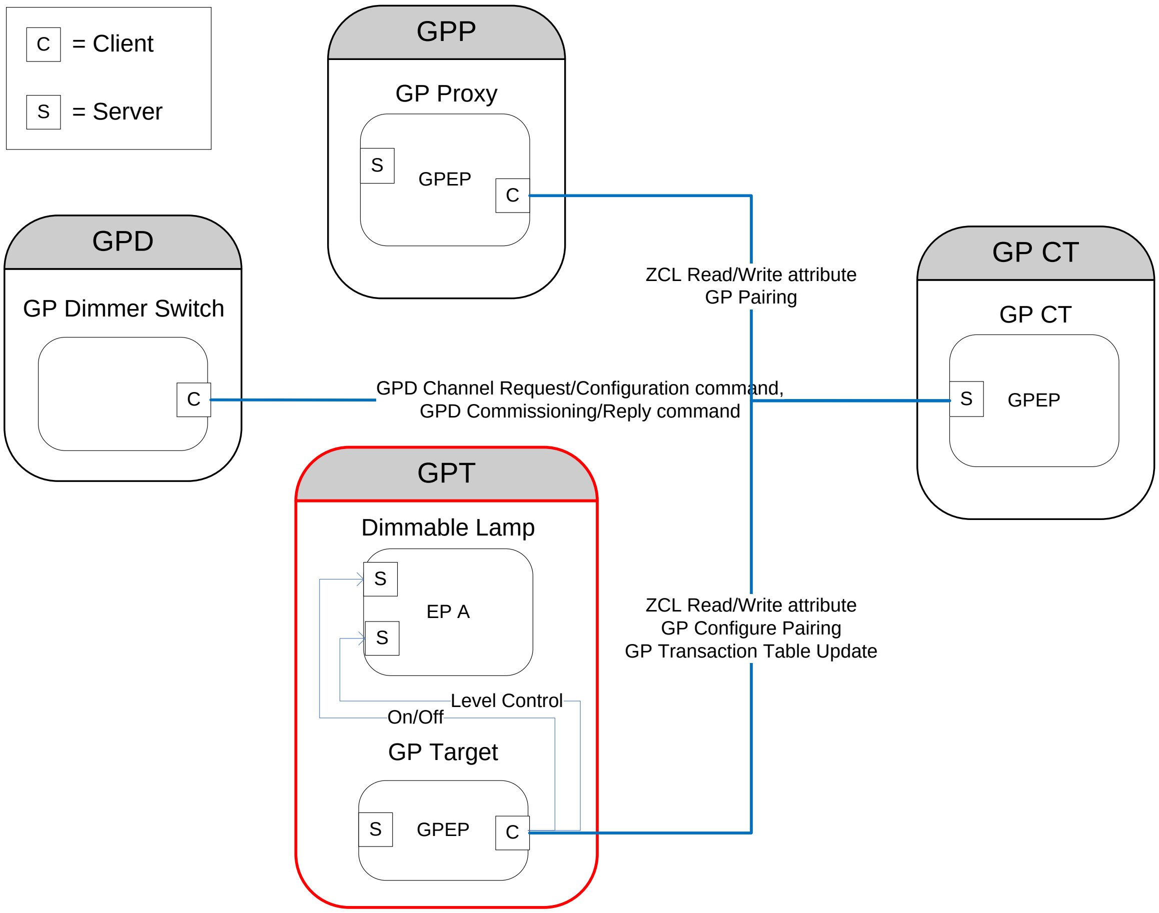


Figure 17 – Example of GP Commissioning Tool device usage

A.3.2.6 GP Proxy minimum device

The functionality supported by the GP Proxy minimum device is defined in Table 21.

Table 21 – Functionality of GP Proxy minimum device

Server side	Client side
Mandatory	
	Selected GreenPower cluster commands (see Table 25) and GP functionality (see Table 24)
	Rx GP stub
Optional	

A GPPm is a regular ZigBee device, in most cases a ZR, which implements on its reserved end point GPEP (see sec. A.3.6.1) the GP Proxy minimum DeviceID (see Table 15) with the selected commands of the client side of the GreenPower cluster (see Table 25) and the reception functionality of the GP stub.

GPPm has the following GP proxy capabilities (see also Table 24):

- Ability to receive any GP frame in direct mode when the GPPm 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 GPS 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.

Note, that the minimum proxy functionality, defined as:

- cGP stub with the corresponding SAPs,
- dGP stub with the corresponding SAPs,
- ability of receiving GP Pairing command and storing the GPD pairing information,
- ability of transmitting, upon reception of GPDF from paired GPD, a GP Notification command in derived groupcast and pre-commissioned groupcast,

is provided by a number of GP infrastructure device types, incl. GPPm, GPCm and GPP.

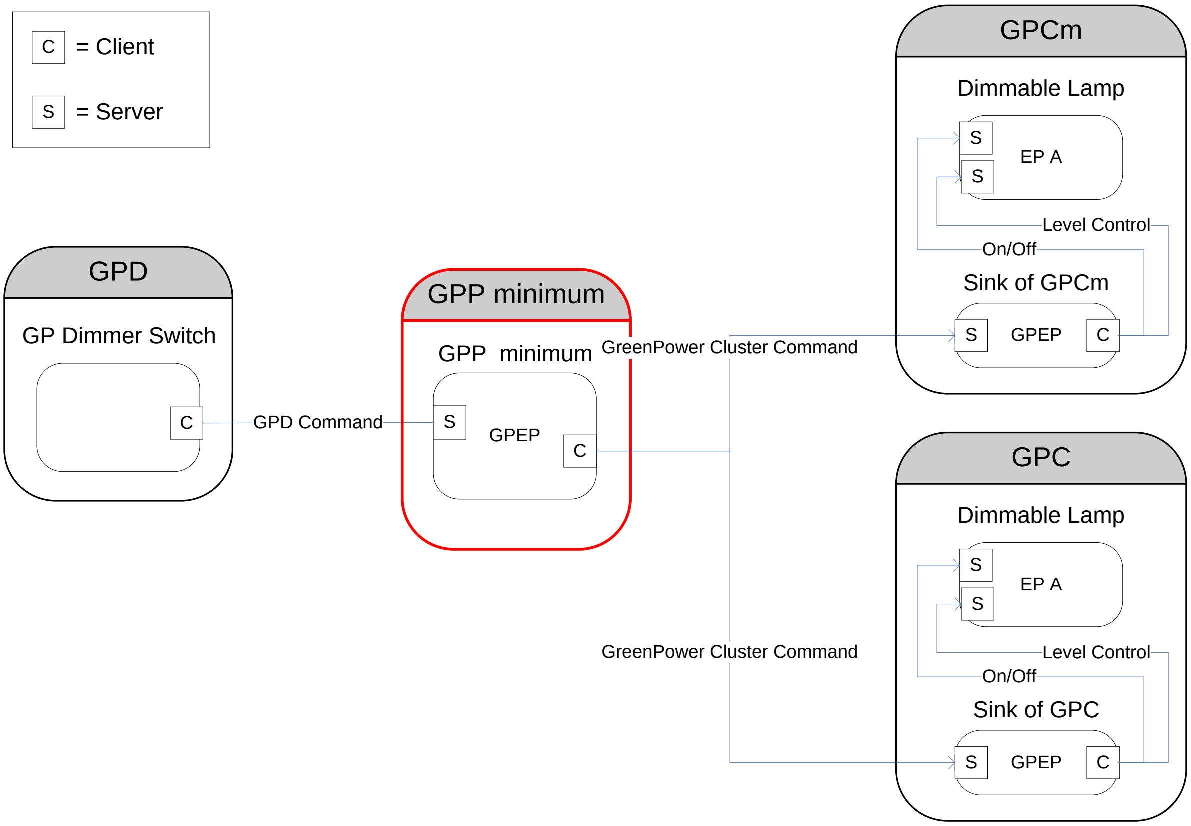


Figure 18 – Example of GP Proxy minimum device usage

A.3.2.7 GP Combo minimum device

The functionality supported by the GP Combo minimum device is defined in Table 22.

Table 22 – Functionality of GP Combo minimum device

Server side	Client side
Mandatory	
Selected GreenPower cluster commands (see Table 25) and GP functionality (see Table 24)	Selected GreenPower cluster commands (see Table 25) and GP functionality (see Table 24)
	Rx GP stub
Optional	
Tx GP stub	

A GPCm implements a subset of the GPC functionality, as depicted in Table 25 and Table 24.

A GPCm device ***shall not*** implement the client attributes of the GreenPower cluster. The data required for the proxy function (i.e. for forwarding to another device), such as groupcast mode, group address, alias, radius and security settings, is available in the Sink Table attribute. This is even possible if there is no pairing to the local endpoints of this GPCm; see A.3.5.2.4 for details.

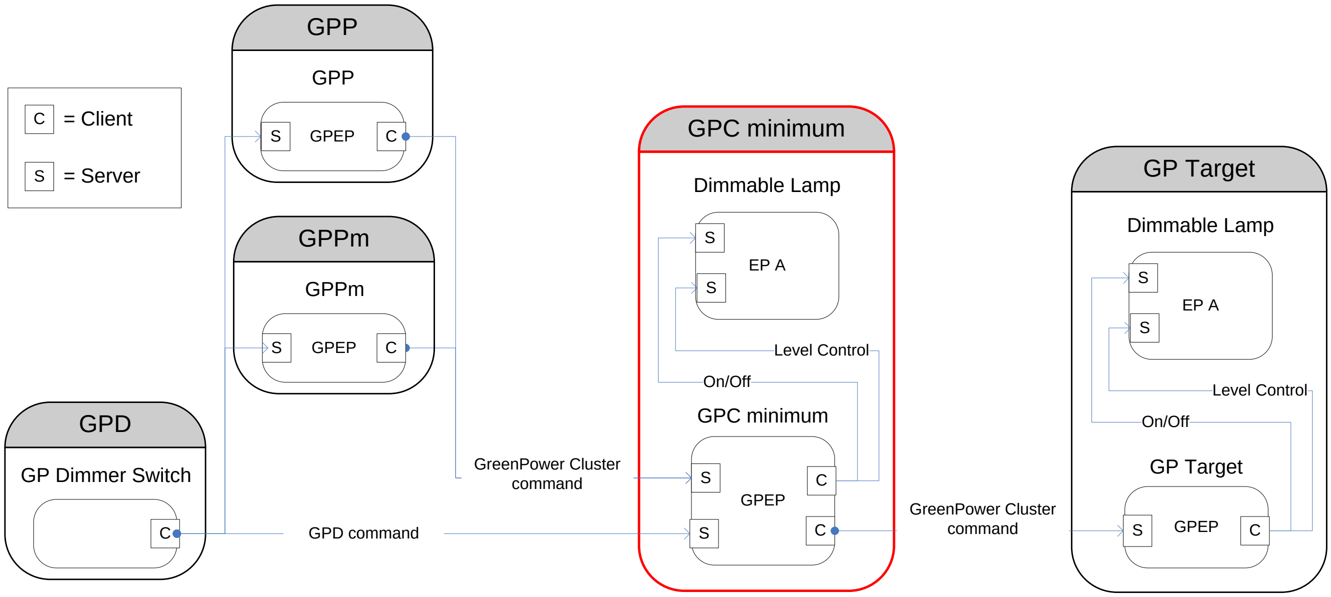


Figure 19 – Example of GP Combo minimum device usage

A.3.2.8 GP functionality

The GP specification defines various functionality to optimize Green Power operation (see sec. A.3.3.2.7 and A.3.4.2.7). Table 23 provides an overview of the GP objects (commands, attributes, primitives, functions, etc.) utilized by this functionality, and is meant as implementation support only.

Table 23 – GP functionality: required commands and functions

Functionality	Elements in a Proxy	Elements in a Sink
Common elements	GP stub for Rx (incl. security), GPEP duplicate filtering , GPEP security check, Proxy Table, Rx GP Pairing	GPEP duplicate filtering, GPEP security check, Sink Table, GPD command execution
Direct communication (reception of GPDF via GP stub)		GP stub for Rx (incl. security)
Derived groupcast communication	Tx groupcast GP Notification with alias	Rx groupcast GP Notification
Pre-commissioned groupcast communication	Tx groupcast GP Notification with alias	Rx groupcast GP Notification, GPEP duplicate filtering, GPEP security check, Sink Table, Rx GP Pairing
Unicast communication	gppTunnelingDelay, Tx GP Tunneling Stop with alias, Rx GP Tunneling Stop, drop own scheduled transmission on Rx GP Stop Tunneling, Tx unicast GP Notification without alias, Rx GP Notification Response, retry	Rx unicast GP Notification, Tx GP Notification Response
Lightweight unicast communication	Rx GP Pairing, Tx unicast GP Notification without alias,	Tx GP Pairing, TempMaster election, Rx unicast GP Notification,
Single-hop (in sink's range) bidirectional operation	N/A	GP stub for Rx (incl. security), GP stub for Tx (incl. security), gpTxQueue
Multi-hop (Proxy-based) bidirectional operation	GP stub for Tx (incl. security), gpTxQueue, Tx GP Notification without alias, Rx GP Notification without alias, drop own scheduled transmission on Rx GP Notification with better TempMaster, Rx GP Response,	Rx GP Notification, TempMaster election, Tx GP Response
Proxy Table maintenance (for GPD mobility and GPP robustness)	Tx broadcast GP Notification, Tx GP Pairing Search, Rx GP Pairing, passive discovery, active discovery,	Rx broadcast GP Notification, Rx GP Pairing Search, Tx GP Pairing
Single-hop (in sink's range) commissioning	Rx GP Pairing	Commissioning mode, Rx GPD Commissioning command, Tx GP Pairing, Tx Device_annce for the alias, Rx GPD Decommissioning command
Single-hop (in sink's range) bidirectional commissioning	Rx GP Pairing	Commissioning mode, gpTxQueue, GP stub for Rx, GP stub for Tx, (GP stub security), GPDF format for Channel Request/Configuration, GPDF Commissioning/Commissioning Reply, GPD application functionality matching, GPDF Commissioning Success, Tx GP Pairing, Tx Device_annce for the alias, Rx GPD Decommissioning command
Multi-hop (Proxy-based) commissioning	commissioning mode, Rx GP Proxy commissioning Mode, Tx GP Commissioning Notification with alias, Tx Device_annce for the alias	Commissioning mode, Tx GP Proxy Commissioning Mode, Rx GP Commissioning Notification with alias, GPD application functionality matching, Tx GP Pairing, Rx GPD Decommissioning command
Multi-hop (Proxy-based) bidirectional commissioning	commissioning mode, Rx GP Proxy commissioning Mode, Tx GP Commissioning notification without alias, Rx GP Commissioning Notification, drop own scheduled transmission on Rx GP Notification with better TempMaster, Tx Device_annce for the alias, Rx GP Response, gpTxQueue, GP stub for Tx (incl. security), changed GPDF format for Channel Request/Configuration,	Commissioning mode, Tx GP Proxy Commissioning Mode, Rx GP Commissioning Notification without alias, Temp Master election, Rx tunneled GPD Channel and GPD Commissioning and GPD Success command, Tx GP Response with GPD Channel Configuration and GPD Commissioning Reply command, GPD application functionality matching, Tx GP Pairing, Rx GPD Decommissioning command
CT-based commissioning	Read access to Proxy Table, Write access to Proxy Table/Rx GP Pairing	Read access to Sink Table, Write access to Sink Table/Rx GP Pairing/Rx GP Pairing Configuration, OPTIONAL: Translation Table, Rx Translation Table Update, Rx Translation Table Request, Tx Translation Table Response

Maintenance of GPD (deliver channel/key during operation)	GP stub for Tx (incl. security), gpTxQueue, Tx GP Notification without alias, Rx GP Notification without alias, drop own scheduled transmission on Rx GP Notification with better Temp-Master, Rx GP Response, changed GPDF format for Channel Configuration,	Rx GP Notification without alias, Temp Master election, Tx GP Response with GPD Channel Configuration and GPD Commissioning Reply command
gpdSecurityLevel = 0b00	gpdSecurityLevel = 0b00 frame processing	For direct communication: gpdSecurityLevel = 0b01 frame processing in the GP stub
gpdSecurityLevel = 0b01	gpdSecurityLevel = 0b01 frame processing, Nonce recovery, gppSecurityWindow	For direct communication: gpdSecurityLevel = 0b01 frame processing in the GP stub, nonce recovery
gpdSecurityLevel = 0b10	gpdSecurityLevel = 0b10 frame processing,	For direct communication: gpdSecurityLevel = 0b10 frame processing in the GP stub
gpdSecurityLevel = 0b11	gpdSecurityLevel = 0b11 frame processing,	For direct communication: gpdSecurityLevel = 0b11 frame processing in the GP stub
Sink Table-based groupcast forwarding	N/A	Tx GP Pairing Configuration, Rx GP Pairing Configuration, OPTIONAL: Translation Table, Rx Translation Table Update, Rx Translation Table Request, Tx Translation Table Response
Translation Table	N/A	Translation Table, Rx Translation Table Update, Rx Translation Table Request, Tx Translation Table Response
GPD IEEE address	Proxy Table and all GreenPower cluster commands (except for GP Proxy Commissioning Mode command) carry IEEE address instead of SrcID	Sink Table, GPD Command Translation Table and all GreenPower cluster commands (except for GP Proxy Commissioning Mode and Translation Table Request command) carry IEEE address instead of SrcID

Selected of the above functionality is indicated in the GreenPower cluster attributes (see Table 30). The GP functionality implemented by a particular infrastructure device is indicated in its *gppFunctionality* / *gpsFunctionality* attribute. The functionality implemented by a particular device can be enabled / disabled by setting the corresponding sub-field of the *gppActiveFunctionality*/*gpsActiveFunctionality* attribute to 0b0.

A.3.2.9 GP functionality support per GP infrastructure device

Table 24 summarizes GP commands support required for each device type of GP infrastructure device role.

The following notations are used to indicate the requirement status:

- M Mandatory
- 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

Table 24 – GP functionality support by GP infrastructure device

Functionality Name	Implementation				
	GPP (standalone or of GPC)	GPPm (standalone)	GPT (standalone GPS)	GPS (of GPC or GPT+)	GPS of GPCm
Common elements	M	M	M	M	M
Direct communication (reception of GPDF via GP stub)	M	M	X	M	M
Derived groupcast communication	M	M	O.1	O.2	O
Pre-commissioned groupcast communication	M	M	O.1 (M if derived groupcast supported)	O.2 (M if derived groupcast supported)	M
Unicast communication	M	O	O.1	O.2	N/A
Lightweight unicast communication	O	O	O.1	O.2	N/A
Single-hop (in sink's range) bidirectional operation	N/A	N/A	X	O	O
Multi-hop (Proxy-based) bidirectional operation	M	O	O	O	O
Proxy Table maintenance (active and passive, for GPD mobility and GPP robustness)	O	O	M	M	O
Single-hop (in sink's range) commissioning	N/A	N/A	N/A	M	M
Single-hop (in sink's range) bidirectional commissioning	N/A	N/A	N/A	M	M
Multi-hop (Proxy-based) commissioning	M	O	M	O	O
Multi-hop (Proxy-based) bidirectional commissioning	M	O	M	O	O
CT-based commissioning	M	O	O	O	M
Maintenance of GPD (deliver channel/key during operation)	M	O	O	O	O
gpdSecurityLevel = 0b00	M	M	O.3	O.4	O.5
gpdSecurityLevel = 0b01	M	O	O.3	O.4	O.5
gpdSecurityLevel = 0b10	M	M	O.3	O.4	O.5
gpdSecurityLevel = 0b11	M	O	O.3	O.4	O.5
Sink Table-based groupcast forwarding	N/A	N/A	O	O	M
Translation Table	N/A	N.A	O	O	O
GPD IEEE address	O	O	O	O	O

A.3.2.10 GP command support per GP infrastructure device

Table 25 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
• 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

Table 25 – GreenPower cluster: command implementation by GP infrastructure device

Command Name	Implementation							
	GPP (standalone or of GPC)		GPPm		GPS (of GPT/GPT+ or GPC)		GPCm	
	Tx	Rx	Tx	Rx	Tx	Rx	Tx	Rx
GP Notification	Groupcast: M Unicast: M Broadcast: O	Groupcast: M Unicast: N/A Broadcast: O	Groupcast: M Unicast: O Broadcast: O	Groupcast: O Unicast: N/A Broadcast: O	N/A	M (at least one of groupcast/unicast) Broadcast: O	Groupcast 0b01: O Groupcast 0b10: M Unicast: N/A Broadcast: O	Groupcast 0b01: O Groupcast 0b10: M Unicast: N/A Broadcast: O
GP Tunneling Stop	M	M	O	O	N/A	N/A	N/A	N/A
GP Pairing Search	O	O	O	O	N/A	M	O	O
GP Notification Response	N/A	M	N/A	O	O (M for GPS with *CommunicationMode*=0b00 and 0b11)	N/A	N/A	N/A
GP Pairing	N/A	M	N/A	M	M	N/A	M	N/A
GP Proxy Commissioning Mode	N/A	M	N/A	O	O (M for GPT)	O	O	O
GP Commissioning Notification	M	M	O	O	N/A	O (M if Tx GP Proxy Commissioning Mode)	O	O
GP Response	N/A	M	N/A	O	O	N/A	O	O
GP Translation Table Update command	N/A	N/A	N/A	N/A	N/A	O	M	M
GP Translation Table Request	N/A	N/A	N/A	N/A	N/A	O	O	O
GP Translation Table Response	N/A	N/A	N/A	N/A	O (M if Rx GP Translation Table Request)	N/A	O	O
GP Pairing Configuration	N/A	N/A	N/A	N/A	O (M for GPS with *CommunicationMode*=0b10)	O (M for GPS with *CommunicationMode*=0b10)	M	M

A GP infrastructure device ***shall*** silently drop any received GP command it does not support. 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 GreenPower cluster contains the attributes shown in Table 26. The M/O column indicates if it is mandatory or optional to support this attribute.

Table 26 applies to GPS devices.

Table 26 – 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	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 GPS
0x0003	gpsCommissioningExitMode	8-bit bitmap	N/A	R/W	0x02	M	Conditions for the GPS to exit the commissioning mode
0x0004	gpsCommissioningWindow	unsigned 16-bit integer	Any valid	R/W	0x0005	O	Default duration of the Commissioning window duration, in seconds, as requested by this GPS
0x0005	*gpsSecurityLevel*	8-bit bitmap	N/A	R/W	0x01	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 GPS
0x0007	*gpsActiveFunctionality*	24-bit bitmap	N/A	R	0xffffff	M	The optional GP functionality supported by this GPS that is active
0x0008-0x000f	Reserved for other attributes of GreenPower 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 GreenPower cluster (see Table 32)						
0x0030-0xffff	Reserved						

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 GPS.

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

A.3.3.2.2 Sink Table

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

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 command and GP Pairing Configuration command of the GreenPower cluster can be used for that purpose.

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 set 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 set of octets, formatted as shown in Table 27. 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* parameter:
  - *ApplicationID* = 0b000 indicates the GPD_ID parameter has the length of 4B and contains the SrcID.
  - *ApplicationID* = 0b010 indicates the GPD_ID parameter has the length of 8B and contains IEEE address.
  - All values of *ApplicationID* other than 0b000 and 0b010 are reserved in the current version of the GreenPower cluster specification.
- The *Group list* parameter:
  - ***shall*** only be included if *Communication mode* 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 28;
  - ***shall*** be completely omitted otherwise (i.e. event 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,
  - be present, if *SecurityUse* = 0b0 and *MAC sequence number capabilities* = 0b1, and carry the value of the GPD's MAC sequence number in 1 LSB, pre-padded with 0x00;
  - be omitted if *SecurityUse* = 0b0 and *MAC sequence number capabilities* = 0b0.

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 27.

The Sink Table ***shall*** be persistently stored.

Table 27 – 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
DeviceID	8-bit enumeration	Any valid (see Table 51)	N/A	M	The DeviceID for this GPD
Group list	set 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 *Options* parameter)

A.3.3.2.2.1 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

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. All values of *ApplicationID* other than 0b000 and 0b010 are reserved in the current version of the GreenPower 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 29.

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 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 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.3.3.2.2.3 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 28.

Table 28 – 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 *Communication mode* 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.3.3.2.2.4 Groupcast radius parameter

The *Groupcast radius* contains the intended radius for the groupcast communication, in number of hops. The default value of 0xff indicates indefinite, i.e. unlimited groupcast.

If *Groupcast radius* parameter is set to a value 0xff and another value is received, the new value ***shall*** be kept. If *Groupcast radius* parameter is set to a value other than 0xff 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.3.3.2.2.5 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	SecurityKey-Type	Reserved

Figure 21 – Format of the Security Options parameter

If *SecurityLevel* is 0b00 or if the *SecurityKeyType* has value 0b011 (GPD 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 GPS; the last two bits can take values as defined in Table 29.

Table 29 – Values of *gpsCommunicationMode* attribute

Value	Description
0b00	unicast forwarding of the GP Notification command both by proxies supporting the lightweight unicast functionality (without observing of *gppTunnelingDelay* and without the transmission/reception of the GP Tunneling Stop command), and by proxies supporting the full unicast functionality (with observing of *gppTunnelingDelay* and with the transmission/reception of the GP Tunneling Stop command)
0b01	groupcast forwarding of the GP Notification command to DGroupID (see A.3.6.1.4))
0b10	groupcast forwarding of the GP Notification command to pre-commissioned GroupID
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)

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, preferably those 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 GPS. 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 GPS accepts pairing changes (additions/removals).

A.3.3.2.6 gpsSecurityLevel attribute

The *gpsSecurityLevel* attribute contains the minimum security level this GPS requires the paired GPDs to support.

It can take values as defined in Table 13.

A.3.3.2.7 gpsFunctionality attribute

The *gpsFunctionality* attribute indicates support of the GP functionality by this device. Any 1-bit subfield 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 30.

Table 30 – Format of the gpsFunctionality attribute

Indication	Functionality
b0	GP feature
b1	Direct communication (reception of GPDF via GP stub)
b2	Derived groupcast communication
b3	Pre-commissioned groupcast communication
b4	Unicast communication
b5	Lightweight unicast communication
b6	Single-hop (in sink’s range) bidirectional operation
b7	Multi-hop (Proxy-based) bidirectional operation
b8	Proxy Table maintenance (active and passive, for GPD mobility and GPP robustness)
b9	Single-hop (in sink’s range) commissioning (unidirectional and bidirectional)

b10	Multi-hop (Proxy-based) commissioning (unidirectional and bidirectional)
b11	CT-based commissioning
b12	Maintenance of GPD (deliver channel/key during operation)
b13	gpdSecurityLevel = 0b00
b14	gpdSecurityLevel = 0b01
b15	gpdSecurityLevel = 0b10
b16	gpdSecurityLevel = 0b11
b17	Sink Table-based groupcast forwarding
b18	Translation Table
b19	GPD IEEE address
b20 – b23	Reserved

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 Table 31.

Table 31 – 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 GPEP ***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 GreenPower cluster contain the attributes shown in Table 32. The M/O column indicates if it is mandatory or optional to support this attribute.

Table 32 – Attributes shared by client and server of the GreenPower cluster

ID	Name	Type	Range	Access	Default	M/O	Description
0x0020	*gpSharedSecurityKeyType*	8-bit bitmap	0x00-0x08	R/W	0b011	O (M if *SecurityLevel* 0b01-0b11 supported)	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	O (M if *SecurityLevel* 0b01-0b11 supported)	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	‘ZigBee Alliance09’	O (M if *SecurityLevel* 0b01-0b11 supported)	The security key to be used to encrypt the key exchanged with the GPD
0x0023-0x002f	Reserved for other attributes shared by GPS and GPP						

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 14: 0b000 (no key), 0b001 (NWK key), 0b010 (GP group key), 0b011 (NWK-key derived GP group key) and 0b111 (Derived individual GPD key).

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 33.

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 25.

Table 33 – GreenPower cluster: server side: commands received

Command ID	Command Name	Command Description	Link
0x00	GP Notification	From GPP to GPS to tunnel GP frame.	A.3.3.4.1
0x01	GP Pairing Search	From GPP to GPSs 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 GPP to neighbor GPPs to indicate GP Notification sent in unicast mode.	A.3.4.4.1
0x04	GP Commissioning Notification	From GPP to GPS to tunnel GPD commissioning data.	A.3.3.4.3
0x05	Reserved		
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-0xff	Reserved		

A.3.3.4.1 GP Notification command

The payload of the GP Notification command shall be formatted as illustrated in Figure 23.

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

Figure 23 – Format of the GP Notification command

Bits: 0..2	3	4	5	6-7	8-10	11	12	13-15
ApplicationID	Also Unicast	Also Derived Group	Also Commissioned Group	SecurityLevel	SecurityKeyType	AppointTempMaster	gppTxQueueFull	Reserved

Figure 24 – Format of the Options field of the GP 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. *ApplicationID* = 0b010 indicates the GPD_ID field has the length of 8B and contains the GPD IEEE address. All values of *ApplicationID* other than 0b000 and 0b010 are reserved in the current version of the GreenPower cluster specification.

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

The *SecurityLevel* sub-field has value copied from the received GPDF and can take values as specified in A.1.5.3.2.

The S*ecurityKeyType* 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 A.1.5.3.3.

The *AppointTempMaster* sub-field, when set to 0b1, indicates that the fields *GPP short address* and *GPP distance* are present.

The *gppTxQueueFull* sub-field indicates whether the GPP 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.

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 *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 0b01- 0b11, it carries the complete 4B frame counter that was successfully used for the security processing of the received GPDF.

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.

A.3.3.4.1.1 When generated

The GP Notification command is generated by the GPP to forward the received Data GPDF to the paired GPSs.

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 GPP.

A.3.3.4.2 GP Pairing Search command

The payload of the GP Pairing Search command shall be formatted as illustrated in Figure 25.

Octets	2	4/8
Data Type	16-bit bitmap	unsigned 32-bit integer/IEEE address
Field Name	Options	GPD ID

Figure 25 – Format of the GP Pairing Search command

The *Options* field of the GP Pairing Search command is formatted as shown in Figure 26.

Bits: 0..2	3	4	5	6	7	8..15
ApplicationID	Request Unicast Sinks	Request Derived Groupcast Sinks	Request Commissioned groupcast sinks	Request GPD Security Frame Counter	Request GPD Security key	Reserved

Figure 26 – 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. *ApplicationID* = 0b010 indicates the GPD_ID field has the length of 8B and contains the GPD IEEE address. All values of *ApplicationID* other than 0b000 and 0b010 are reserved in the current version of the GreenPower cluster specification.

The *RequestUnicastSinks* sub-field shall be set to 0b1, if the proxy requests pairing information on unicast sinks for the GPD specified in *GPD ID* 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 requests 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 *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 GPP 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 GPP 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 27.

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

Figure 27 – Format of the GP Commissioning Notification command

The *Options* field of the GP Commissioning Notification command shall be formatted as shown in Figure 28.

Bits: 0..2	3	4..5	6..8	9	10..15
ApplicationID	Appoint TempMaster	SecurityLevel	SecurityKeyType	Security processing failed	Reserved

Figure 28 – 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. *ApplicationID* = 0b010 indicates the GPD_ID field has the length of 8B and contains the GPD IEEE address. All values of *ApplicationID* other than 0b000 and 0b010 are reserved in the current version of the GreenPower cluster specification.

The *AppointTempMaster* sub-field allows the GPP to request GPS to select a GPP to forward Commissioning Reply GPDF to this GPD. If it is set to 0b1, then the fields *GPP short address* and *GPP distance*, carrying the NWK address of the GPP and the distance to the GPD, shall be included, otherwise not.

SecurityLevel is copied from the *SecurityLevel* sub-field of the *Extended NWK Frame Control* field of the received GPDF, set to 0b00, if not present there.

SecurityKeyType corresponds to the type of the key successfully used for GPDF processing.

Security processing failed sub-field shall be set to 0b1, if the Commissioning GPDF was protected, but the security check failed.

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.

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, or if *Security processing failed* sub-field of the *Options* field of the GP Commissioning Notification is set to 0b1, 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 0b01-0b11 and *Security processing failed* sub-field is set to 0b0, it carries the complete 4B frame counter that was successfully used for the security processing 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.

If the *SecurityLevel* sub-field of the *Options* field is set 0b00 – 0b10 or if *SecurityLevel* sub-field of the

Options field is set to 0b11 and the *Security processing failed* 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 *Security processing failed* 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 *MIC* field ***shall*** only be present if the *Security processing failed* sub-field is set to 0b1.

A.3.3.4.3.1 When generated

The GP Commissioning Notification command is used by the GPP in commissioning mode to forward commissioning data to the GPS(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.

A.3.3.4.4 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 46). The payload of the GP Translation Table Update command shall be formatted as illustrated in Figure 29.

Octets	2	4/8	Variable	…	Variable
Data Type	16-bit bitmap	unsigned 32-bit integer/IEEE address	Variable	…	Variable
Field Name	Options	GPD ID	Translation 1	…	Translation N

Figure 29 – 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 30.

Bits: 0..2	3..4	5..7	8..15
ApplicationID	Action	Number of Translations	Reserved

Figure 30 – 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. *ApplicationID* = 0b010 indicates the GPD_ID field has the length of 8B and contains the GPD IEEE address. All values of *ApplicationID* other than 0b000 and 0b010 are reserved in the current version of the GreenPower cluster specification.

The *Action* sub-field of the *Options* field can take the values as specified in Table 34.

Table 34 – 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 GPS, 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 ***may*** be returned. If the *Index* field has the value of 0xff, the GPS ***shall*** choose any free entry. Already existing translation entry for the same (GPD ID, GPD CommandID, EndPoint, Profile, Cluster) quintuple present in the GPS Command Translation Table, if any, ***shall not*** be affected.

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 GPS, in the entry number as specified by the *Index* field. Translation entry(s) for the same (GPD ID, GPD CommandID, EndPoint, Profile, Cluster) quintuple stored in the GPS Command Translation Table under different *Index* value, if any, ***shall not*** be affected by this command.

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 GPS. The values of the remaining sub-fields of the Translation field are ignored. If the *Index* field is set to 0xff, all entries for this GPD ID shall be removed.

The *Number of Translations* indicates how many Translation fields are included in the command. 0b000 indicates none.

The *Translation* field of the GP Translation Table Update command is formatted as illustrated in Figure 31.

Octets	1	1	1	2	2	1	1	0/Variable
Data Type	unsigned 8-bit integer	unsigned 8-bit integer	unsigned 8-bit integer	unsigned 16-bit integer	unsigned 16-bit integer	unsigned 8-bit integer	unsigned 8-bit integer	Set of unsigned 8-bit integer
Field Name	Index	GPD Command ID	EndPoint	Profile	Cluster	ZigBee Command ID	ZigBee Command payload length	ZigBee Command payload

Figure 31 – Format of the Translation field of the GP Translation Table Update command

The *Index* field determines the Translation Table entry. The first entry has the *Index* value of 0.

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 GPS 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. Otherwise, a fixed payload for the ZigBee command is provided, of the *ZigBee Command payload length*.

A.3.3.4.4.1 When generated

This command is generated to configure the GPD Command Translation Table.

A.3.3.4.4.2 Effect on Receipt

On receipt of this command, a GPS 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 32.

Octets	**1**
Data Type	8-bit bitmap
Field Name	Start index

Figure 32 – 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 GPS shall send a GP Translation Table Response command.

A.3.3.4.6 GP Pairing Configuration command

The GP Pairing Configuration command shall be formatted as illustrated in Figure 33 and Figure 34.

Octets	1	2	4/8	1	0/Variable	0/2
Data Type	Unsigned 8-bit integer	16-bit bitmap	Unsigned 32-bit integer/IEEE address	8-bit enumeration	Unsigned 16-bit integer	Unsigned 16-bit integer
Field Name	Actions	Options	GPD ID	DeviceID	GroupList	GPD Assigned Alias

Figure 33 – 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	Set of Unsigned 8-bit integer
Forwarding Radius	Security Options	GPD security frame counter	GPD security Key	Number of paired endpoints	Paired endpoints

Figure 34 – Format of the GP Pairing Configuration command (part 2)

The *Actions* field is formatted as shown in Figure 35.

Bits: 0-2	3	4-7
Action	Send GP Pairing	Reserved

Figure 35 – 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 35.

Table 35 – 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.
0b101-0b111	Reserved

The *Send GP Pairing* sub-field, if set to 0b1 indicates that the receiving GPS is requested to send GP Pairing command upon completing the handling of GP Pairing Configuration. If set to 0b0, it indicates that the receiving GPS ***shall not*** send GP Pairing command upon completing the handling of the GP Pairing Configuration command.

All the fields Options, GPDID, DeviceID, GroupList, GPD Assigned Alias, Forwarding 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 GPS 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.

If the *Action* sub-field of the *Actions* field is set to 0b000, only the following fields of the Pairing Configuration command are of importance to the receiving GPS: *Send GP Pairing* sub-field, and if set to 0b1, the *GPD ID*. The other fields of the Pairing Configuration command: *Options*, *DeviceID*, *Pre-commissioned GroupID*, *GPD Assigned Alias*, *Forwarding Radius*, *Security Options*, *GPD security frame counter*, *GPD security Key*, *Number of paired endpoints,* and *Paired endpoints*, if present, are ignored.

If the *Action* sub-field of the *Actions* field is set to 0b100, only the *GPD ID* field of the Pairing Configuration command is of importance to the receiving GPS. The other fields of the Pairing Configuration command: *Options*, *DeviceID*, *GroupList*, *GPD Assigned Alias*, *Forwarding Radius*, *Security Options*, *GPD security frame counter*, *GPD security Key*, *Number of paired endpoints*, and *Paired endpoints*, if present, are ignored.

If the *Action* sub-field of the *Actions* field is set to a 0b011, the following fields of the received Pairing Configuration command are of importance: *CommunicationMode* sub-field of the *Options* field, the *GroupList,* if present, *Number of paired endpoints,* and *Paired endpoints*, if present. The other fields of the received Pairing Configuration command: *DeviceID*, *GPD Assigned Alias*, *Forwarding 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 Pairing Configuration command are of importance.

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.3.3.4.6.1 When Generated

The command is generated to configure the Sink Table of a GPS, to create/update/replace/remove a pairing to a GPD and/or trigger the sending of GP Pairing command.

A.3.3.4.6.2 Effect on Receipt

On receipt of this command, the receiver is informed about the request to modify its Sink Table.

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 25.

Table 36 – GreenPower cluster: server side: commands generated

Command Value	Command Name	Command Description	Link
0x00	GP Notification Response	From GPS to GPP to acknowledge GP Notification received in unicast mode.	A.3.3.5.1
0x01	GP Pairing	From GPS 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 GPS to GPPs in the whole network to indicate commissioning mode	A.3.3.5.3
0x03-0x05	Reserved		
0x06	GP Response	From GPS to selected GPP, 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 – 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 36.

Octets	1	4/8	4
Data Type	8-bit bitmap	unsigned 32-bit integer/IEEE address	Unsigned 32-bit integer
Field Name	Options	GPD ID	GPD security frame counter

Figure 36 – Format of the GP Notification Response command

The *Options* field shall be formatted as shown in Figure 37.

Bits: 0..2	3	4	5..7
ApplicationID	FirstToForward	NoPairing	Reserved

Figure 37 – 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. *ApplicationID* = 0b010 indicates the GPD_ID field has the length of 8B and contains the GPD IEEE address. All values of *ApplicationID* other than 0b000 and 0b010 are reserved in the current version of the GreenPower 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 GPS 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.

The *GPD security frame counter* copied from the GP Notification.

A.3.3.5.1.1 When generated

This command is generated when the GPS acknowledges the reception of 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 GPP is informed about GPS having received a 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 38 and Figure 39.

Octets	3	4/8	0/8	0/2	0/2
Data Type	24-bit bitmap	unsigned 32-bit integer/IEEE address	IEEE address	unsigned 16-bit integer	unsigned 16-bit integer
Field Name	Options	GPD ID	Sink IEEE address	Sink NWK address	Sink GroupID

Figure 38 – 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	Forwarding Radius

Figure 39 – Format of the GP Pairing command (part 2)

The *Options* field of the GP Pairing command shall be formatted as illustrated in Figure 40 and Figure 41.

Bits: 0..2	3	4	5..6	7	8	9..10
ApplicationID	Add Sink	Remove GPD	Communication mode	GPD Fixed	GPD MAC sequence number capabilities	SecurityLevel

Figure 40 – 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	GPD security key present	Assigned Alias present	Forwarding Radius present	Reserved

Figure 41 – Format of the Options field of the GP Pairing 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. *ApplicationID* = 0b010 indicates the GPD_ID field has the length of 8B and contains the GPD IEEE address. All values of *ApplicationID* other than 0b000 and 0b010 are reserved in the current version of the GreenPower 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*, *Assigned alias, Forwarding 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 *Communication mode* sub-field defines the communication mode requested by the GPS, and can take values as defined in Table 29.

The *GPDfixed* sub-field and *GPD MAC sequence number capabilities* sub-field is copied from the corresponding *FixedLocation* field 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.
The *GPDsecurityFrameCounter* field ***shall*** be present whenever the *AddSink* sub-field of the *Options* field is set to 0b1; independent of the security level. If the *SecurityLevel* sub-field is set to 0b01-0b11, the *GPDsecurityFrameCounter* 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 0b1, the 1LSB of the *GPDsecurityFrameCounter* carries the current value of the GPD MAC sequence number field from the Sink Table entry corresponding to the *GPD ID*; the remaining octets of the *GPDsecurityFrameCounter* field are set to 0x00. If the *SecurityLevel* is 0b00 and the *GPD MAC sequence number capabilities* sub-field is set to 0b0, the *GPDsecurityFrameCounter* shall be set to 0b00000000.

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 *Forwarding Radius present* sub-field, if set to 0b1, indicates that the *Forwarding Radius* field is present and carries the *Forwarding Radius* value to be used as value of the radius in the groupcast forwarding of the GPDF packet. If the *Forwarding Radius* field is not present, and a new Proxy Table entry is to be created, the default value of 0xff ***shall*** be used. The value 0xff indicates unlimited radius.

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 presence of the addressing fields (*SinkIEEEaddress, SinkNWKaddress,* and *SinkGroupID*) is indicated by the sub-fields *RemoveGPD* and the *Communication mode* of the *Options* field, as shown in Table 37 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 unicast communication mode is requested. The *SinkGroupID* field is only present, if one of the groupcast communication modes is requested.

Table 37 – 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 GPS originating the GP Pairing command.

The *SinkGroupID* field, if present, carries the GroupID the GPS originating the GP Pairing command is member of.

The *Disable default response* sub-field of the *Frame Control Field* of the ZCL header shall be set to 0b1.

A.3.3.5.2.1 When generated

The GP Pairing command is generated by the GPS 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 42.

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 42 – Format of the GP Proxy Commissioning Mode command

The *Options* field shall be formatted as shown in Figure 43.

Bits: 0	1-3	4	5-7
Action	Exit mode	Channel present	Reserved

Figure 43 – 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.

The *Exit mode* sub-field shall be formatted as shown in Figure 22. 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.

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, 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.

The *CommissioningWindow* field shall be present, if the *On Commissioning Window expiration* flag of the *Exit mode* sub-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.

The *Disable default response* sub-field of the *Frame Control Field* of the ZCL header shall be set to 0b1.

A.3.3.5.3.1 When generated

This command is generated when the GPS wishes to instruct the GPPs to enter/exit commissioning mode. The GP Proxy Commissioning Mode command shall be 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.

A.3.3.5.4 GP Response command

The payload of the GP Response command shall be formatted as illustrated in Figure 44.

Octets	1	2	1	4/8	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	Octet string
Field Name	Options	TempMaster short address	TempMaster Tx channel	GPD ID	GPD Comman-dID	GPD Command payload

Figure 44 – Format of the GP Response command

The *Options* shall be formatted as shown in Figure 46.

Bits: 0..2	**3..7**
ApplicationID	Reserved

Figure 45 – 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 is identified by 4B SrcID, which is stored in the 4LSB of the *GPD ID* field; the 4MSB of the *GPD ID* field contain “0x00”. *ApplicationID = 0b010* indicates the

GPD is identified by 8B IEEE address. All values of *ApplicationID* other than 0b000 and 0b010 are reserved in the current version of the GreenPower cluster specification.

The *TempMaster short address* field indicates the address of the GPP 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 46.

Bits: 0-3	4-7
Transmit channel	Reserved

Figure 46 – Format of the GPP Tx Channel field of the GP Response command

When the *TempMaster short address* field is set to 0xffff, the *TempMaster Tx Channel* field shall be ignored. The *TransmitChannel* sub-field of the *TempMasterTxChannel* field can carry the value of the operational channel.

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 *FrameType*, 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.

A.3.3.5.4.1 When generated

This command is generated when GPS 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 47.

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 47 – Format of the GP Translation Table Response command

The *Status* field can take the values of SUCCESS or NOT_SUPPORTED.

The *Options* shall be formatted as shown in Figure 46.

Bits: 0..2	3..7
ApplicationID	Reserved

Figure 48 – 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 has the length of 4B and contains the GPD SrcID. *ApplicationID* = 0b010 indicates the GPD_ID field has the length of 8B and contains the GPD IEEE address. All values of *ApplicationID* other than 0b000 and 0b010 are reserved in the current version of the GreenPower cluster specification.

The *Total number of entries* field specifies the number of entries in the GPD Command Translation Table (see Table 46) of this GPS.

The S*tart index* field specifies the starting index into the GPD Command Translation Table of this GPS 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 49. The entries in the *TranslationTableList* field are ordered by *Index* field value, with the lowest entry being sent first.

Octets	4/8	1	1	2	2	1	1	0/Variable
Data Type	unsigned 32-bit integer/IEEE address	unsigned 8-bit integer	unsigned 8-bit integer	unsigned 16-bit integer	unsigned 16-bit integer	unsigned 8-bit integer	unsigned 8-bit integer	Set of unsigned 8-bit integer
Field Name	GPD ID	GPD Command ID	EndPoint	Profile	Cluster	ZigBee Command ID	ZigBee Command payload length	ZigBee Command payload

Figure 49 – Format of the entry of the TranslationTableList field of the GP Translation Table Response command

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.

A.3.3.5.5.1 When Generated

The GP Translation Table Response command is generated by a GPS on reception of a GP Translation Table Request command.

When the GPD Command Translation Table is empty, the GP Translation Table Response command ***shall*** carry the SUCCESS in the *Status* field and 0x00 in the *Total number of entries* field.

A.3.3.5.5.2 Effect on Receipt

The receiving device gets information on the GPD Command Translation Table of the GPS that sent the command.

A.3.4 Client

A.3.4.1 Dependencies

None.

A.3.4.2 Attributes

The client side of the GreenPower cluster contains the attributes shown in Table 38. Table 38 applies to GPP devices.

Table 38 – 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	0x0a	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 information about pairings between a particular GPD ID and GPSs in the network
0x0012	gppNotification-RetryNumber	unsigned 8-bit integer	0x00-0x05	R/W	0x02	O (M if any *unicast communication* functionality supported)	Number of unicast GP Notification retries on lack of GP Notification Response
0x0013	gppNotification-RetryTimer	unsigned 8-bit integer	0x00 – 0xff	R/W	0x64	O (M if any *unicast communication* functionality supported)	Time in ms between unicast GP Notification retries on lack of GP Notification Response
0x0014	gppMaxSearch-Counter	Unsigned 8-bit integer	Any valid	R/W	0x0a	O	The frequency of sink re-discovery for inactive Proxy Table entries
0x0015	gppBlockedGPDID	Long octet string	N/A	R	0x0000	O	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 GPP
0x0017	gppActiveFunc-tionality	24-bit bitmap	N/A	R	0xffffff	M	The optional GP functionality supported by this GPP that is active
0x0018 - 0x001f	Reserved for further GreenPower cluster client side attributes						
0x0020 - 0x002f	Attributes shared by GPP and GPS, as defined in Table 26						

0x0030 - 0xffff	Reserved

A.3.4.2.1 gppMaxProxyTableEntries attribute

Maximum number of Proxy Table entries this node can hold.

A.3.4.2.2 Proxy Table attribute

The Proxy Table attribute contains the information on GPDs active in the system and the corresponding GPSs.

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 GreenPower cluster can be used for that purpose.

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 set 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 39. 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* parameter:
  - *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.
  - All values of *ApplicationID* other than 0b000 and 0b010 are reserved in the current version of the GreenPower 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,
  - be present, if *SecurityUse* = 0b0 and *MAC sequence number capabil*ities = 0b1, and carry the value of the GPD's MAC sequence number in 1 LSB, pre-padded with 0x00;
  - be omitted if *SecurityUse* = 0b0 and *Sequence number capabilities* = 0b0.
- *Sink address list* parameter
  - ***shall*** only be included if *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 40; 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 28;

- ***shall*** be completely omitted otherwise (i.e. event the length octet shall be omitted);

- *Search Counter **shall*** be included if *EntryActive*=0b0 or *EntryValid=0b0* sub-field of the *Options* parameter is set to 0b0, it ***shall*** be omitted otherwise.

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 39.

Table 39 – 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
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)
Sink address list	set of octets	Any valid	0x00	O (M if *Unicast GPS* =0b1)	IEEE and short address of the sink(s) that requires tunneling in unicast communication mode
Sink group list	set 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 - *gppMax-Search-Counter*	0x00	O (M if *EntryActive*=0b0 or *EntryValid=0b0)*	For inactive/invalid entries, allows for Sink re-discovery when Search Counter equals *0*

Each GPP 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 *Sink address list*, (ii) at least 2 entries in the *Sink group list* and (iii) at least 1 entry in the *Sink address list* and at least 1 entry in the *Sink group list*.

A.3.4.2.2.1 Options parameter

The *Options* parameter shall be formatted as shown in Figure 50 and Figure 51.

Bits: 0..2	3	4	5	6	7	8	9
ApplicationID	EntryActive	EntryValid	Sequence number capabilities	Unicast GPS	Derived Group GPS	Commissioned Group GPS	FirstTo-Forward

Figure 50 – Format of the Options parameter of the Proxy Table entry (part 1)

Bits: 10	11	12	13	14	15
InRange	GPD Fixed	HasAllUnicastRoutes	AssignedAlias	SecurityUse	Reserved

Figure 51 – 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. *ApplicationID* = 0b010 indicates the *GPD_ID* parameter has the length of 8B and contains the GPD IEEE address. All values of *ApplicationID* other than 0b000 and 0b010 are reserved in the current version of the GreenPower 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 *Unicast GPS* sub-field, if set to 0b1, indicates that there is at least one GPS paired to this GPD, that require unicast communication mode. Then, *Sink address list* parameter is present.

The *Derived Group GPS* sub-field, if set to 0b1, indicates that there is at least one GPS 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 GPS 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 if this GPP. 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 GPP has active routes to all unicast sinks for this GPD; if set to 0b0, it indicates that at least one 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 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.

A.3.4.2.2.2 GPD Assigned Alias parameter

The *GPD Assigned Alias* parameter, if present, stores the assigned alias NWK source address to be used for this GPD in case of unicast communication GPS, instead of the default alias derived from the GPD ID (sec. A.3.6.3.3).

A.3.4.2.2.3 Security-related parameters

The security-related parameters shall be used exactly as described in A.3.3.2.2.5. The security-related parameters are formatted and to be used as described in A.3.3.2.2.4.

A.3.4.2.2.4 Sink address list parameter

The entries in the *Sink address list* parameter shall have the format as specified in Table 40. It contains the list of paired unicast sinks for this GPD.

Table 40 – Format of entries in the *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.3.4.2.2.5 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 28.

If the *Pre-Commissioned Group GPS* sub-field of the *Options* parameter is set, the *Sink group list* ***should*** be present.

A.3.4.2.2.6 Groupcast radius parameter

The *Groupcast radius* contains the intended radius for the groupcast communication, in number of hops. The default value of 0xff indicates indefinite, i.e. unlimited groupcast.

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

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 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 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 41.

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 41.

Table 41 – 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
Sequence number	Unsigned 8-bit integer	0x00-0xff	0x00	M	The last sequence number observed from this GPD.
Search Counter	Unsigned 8-bit integer	0x00 - *gppMaxSearchCounter*	0x00	M	Allows for Sink rediscovery when Search Counter equals *0*

The *Options* parameter shall be formatted as shown in Figure 52.

Bits: 0..2	3..7
ApplicationID	Reserved

Figure 52 – 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. *ApplicationID* = 0b010 indicates the GPD_ID parameter has the length of 8B and contains the GPD IEEE address. All values of *ApplicationID* other than 0b000 and 0b010 are reserved in the current version of the GreenPower 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 30.

For all GPP, GPPm and proxy functionality of GPC, 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).

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 31.

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 GPEP ***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.

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 25.

Table 42 – GreenPower cluster: client side: commands received

Command ID	Command Name	Command Description	Link
0x00	GP Notification Response	From GPS to GPP to acknowledge GP Notification received in unicast mode.	A.3.3.5.1
0x01	GP Pairing	From GPS to GPP to (de)register for tunneling service or to remove GPD from the network.	A.3.3.5.2
0x02	GP Proxy Commission-ing Mode	From GPS to GPPs in the whole network to indicate commissioning mode.	A.3.3.5.3
0x03-0x05	Reserved		
0x06	GP Response	From GPS to selected GPP, to provide data to be transmitted to Rx-capable GPD.	A.3.3.5.4
0x07	Reserved		
0x08	Reserved		
0x09	Reserved		
0x0a – 0xff	Reserved		

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 25.

Table 43 – GreenPower cluster: client side: commands generated

Command ID	Command Name	Command Description	Link
0x00	GP Notification	From GPP to GPS to tunnel GP frame.	A.3.3.4.1
0x01	GP Pairing Search	From GPP to GPSs 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 GPP to neighbor GPPs to indicate GP Notification sent in unicast mode.	A.3.4.4.1
0x04	GP Commissioning Notification	From GPP to GPS to tunnel GPD commissioning data.	A.3.3.4.3
0x05	Reserved		
0x06 – 0x09	Reserved		
0x0a-0xff	Reserved		

A.3.4.4.1 GP Tunneling Stop command

The payload of the GP Tunneling Stop command shall be formatted as illustrated in Figure 53.

Octets	1	4/8	4	2	1
Data Type	8-bit bitmap	unsigned 32-bit integer/IEEE address	unsigned 32-bit integer	unsigned 16-bit integer	signed 8-bit integer
Field Name	Options	GPD ID	GPD security frame counter	GPP short address	GPP distance

Figure 53 – Format of the GP Tunneling Stop command

The *Options* field of the GP Tunneling Stop command shall be formatted as illustrated in Figure 54.

Bits: 0..2	3	4	5..7
ApplicationID	Also Derived Group	Also Commissioned Group	Reserved

Figure 54 – 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. *ApplicationID* = 0b010 indicates the GPD_ID field has the length of 8B and contains the GPD IEEE address. All values of *ApplicationID* other than 0b000 and 0b010 are reserved in the current version of the GreenPower 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 *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 0b01- 0b11, it carries the complete 4B frame counter that was successfully used for the security processing of the received GPDF.

The fields *GPP address* and *GPP distance* are always present and carry the short address of the originating proxy, and the distance in meters between the GPP and the GPD.

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 GPPs from also forwarding GP Notifications to GPSs requiring unicast communication mode.

A.3.4.4.1.2 Effect on Receipt

On receipt of this command, a device is informed about another GPP forwarding a GPDF.

A.3.5 Green Power operation

A.3.5.1 Overview

The GPPs forward the Data GPDFs from the GPDs to paired GPSs as regular ZigBee messages using the ZCL GreenPower cluster commands.

Each GPS has as part of the GreenPower 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 GPS manages the entries in its Sink Table. Sink Table entry changes for a particular GPD are announced to the GPPs by sending a GP Pairing command. The GPS responds to the GPPs' GP Pairing Search commands requesting missing information on paired GPDs by sending GP Pairing commands.

Each GPS 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 GPS forms the response and sends it to the device it has selected for sending the response to the GPD.

Each GPP has as part of the GreenPower cluster a Proxy Table to store pairing information on the GPDs and the paired GPSs, including the security requirements and communication mode.

The GPP participates in management of pairings at the GPSs, 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 GreenPower cluster GP Commissioning Notification command. On receipt of GP Pairing command frames, the GPP manages the entries in its Proxy Table. The GPP can ask for updates on missing or outdated pairing information by sending GP Pairing Search command.

The GPP is responsible for tunneling the received Data GPDFs of the GPDs for which it has valid pairing information to the paired GPS, as the regular ZigBee messages using the ZCL GreenPower cluster GP Notification command.

The GPP forwards Data GPDF to an RxAfterTx-capable GPD, if requested by GPS as indicated by GP Response command.

A.3.5.2 Description

A.3.5.2.1 Proxy operation

On receipt of GP-SEC.request, the proxy acts as described in sec. A.3.7.2.1.

On receipt of ZigBee Update Device and Device_annce commands with IEEE address other than 0xffffffffffffffff, the GPP 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 A.3.6.3.4, using the Alias NWK source address and a fixed sequence number of 0x00, 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* was 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 GPP updates its Proxy Table, if the entry is active.

If the *RemoveGPD* flag was set to 0b1, the GPP, if it does not support the *Proxy Table maintenance* functionality, ***shall*** remove the Proxy Table entry for that GPD. If the GPP 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* flag was set to 0b0; and the *AddSink* flag was set to 0b0, the GPP removes the GPS address or Sink group address from the corresponding *SinkList*, depending on the setting of the *CommunicationMode* sub-field. If the removed unicast/group sink address is the last in the Sink address list/Sink group list, respectively, and no other sink communication mode is used for this GPD, 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 the Proxy Table entry for that GPD.

If the *RemoveGPD* flag was set to 0b0 and the *AddSink* flag was set to 0b1, the GPP adds the communication mode, if new, and the GPS (group) address, if not already included in the corresponding *SinkList*, and sets the entry to active and valid. If a groupcast sink is being added to a Proxy Table entry, the GPP also adds its GPEP 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 the *Assigned Alias* field is present, the proxy stores it in its Proxy table entry, and sets the corresponding *Options* sub-field.

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 GPS or CT sending the GP Pairing command.

On receipt of a GP Response frame from GPS in groupcast, both in operational and commissioning mode, the GPP checks if its short address matches the value in the *TempMaster short address* field. If yes and also if the GP Response command was sent to this proxy in unicast, the GPP adds the GPDF frame derived from the GP Response frame to its *gpTxQueue* for sending to the indicated GPD ID by calling GP-DATA.request with *Action* parameter set to TRUE, and sets its *FirstToForward* flag for this GPD to 0b1. In case of non-matching GPP address, the GPP drops the current command, removes any previous pending GPDF for this GPD in 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.

On receipt of GP-DATA.indication, the proxy checks the GPDF type and the mode the proxy is in.

If the GPDF carries a GPD Commissioning command or a GPD Decommissioning command and the proxy is not in commissioning mode, the GPD Commissioning command and the GPD Decommissioning is 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/NO_SECURITY, the proxy updates the *Sequence number/Frame counter* field of the Proxy Table 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 failed* 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. If there is an 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), the Proxy sets the *InRange* flag to 0b1. Then, the proxy continues as follows.

If the entry is active and valid then the proxy checks the security level of the received GPDF as follows. The GPP 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 GPP 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 GPP ***shall not*** send GP Tunneling Stop/GP Notification; it ***may*** send GP Pairing Search.

If all the checks succeed, the GPP stores the *Sequence Number / Frame Counter* in the 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 the *RxAfterTx* sub-field of the received GPDF was set to 0b1, the *GPPpresent* sub-field of the *Options* field shall be set to 0b1 and the fields *GPP short address* and *GPP distance* shall be included, and the *gppTxQueueFull* sub-field of the *Options* field set according to the status of this GPP's *gpTxQueue* (i.e., if there is no entry in the *gpTxQueue* for this GPD and the queue is full, it sets the *gppTxQueueFull* sub-field to 0b1, otherwise if it has an entry for this GPD or at least one empty entry, it sets it to 0b0). 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 GPP 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.

For the unicast GP Notification command, the GPP shall further use the following values: NWK Src address and NWK sequence number: GPP's own values (no aliasing), NWK Dst address: GPS short address, APS source and destination end point: GPEP. For the GP Tunneling Stop command the GPP shall use proxy aliasing (see sec. A.3.6.3.3) for NWK Src address and NWK Sequence Number, and local radius (2 hops) 0xFFFD broadcast as NWK Dest address.

For groupcast GP Notification, the GPP shall further use the following values: NWK Src address and NWK Sequence Number: proxy alias (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: GPEP.

The GPP schedules sending of the GP Notification command. If there are any full unicast destinations, also in addition to groupcast destinations, or the *RxAfterTx* flag was set, the sending shall be scheduled after *gppTunnelingDelay* (see section A.3.6.3.1); if there are unicast destinations, the *gppTunnelingDelay* is calculated as for the unicast. If 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 for 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 *GPP distance* field from the received command has a higher value than the distance measured by the receiving proxy on receipt of this GPDF, or is the distance value is equal, if the value in the GPP address field is lower than this proxy’s NWK address. Otherwise, if there are only lightweight unicast destinations and/or groupcast destinations, and the *RxAfterTx* flag was cleared, the sending shall be scheduled after *Dmin* (see section A.3.6.3.1).
On timeout, the GP Tunneling Stop command (if any) ***shall*** be sent first, the remaining commands ***should*** be sent in the following order: the unicast GP Notification(s) (if any), groupcast GP Notification(s) (if any). Upon transmission of unicast GP Notification, the GPP 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 GPS are dropped, and the *FirstToForward* bit in the GPPs Proxy Table entry for this GPD 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 GPP shall remove this GPS from its *SinkAddressList* in the Proxy Table entry for this GPD. If no GP Notification Response command is received after last retry of the unicast GP Notification, the GPP ***may*** ask the ZigBee stack to re-discover the route to this unicast GPS. It ***may*** pro-actively clear the *HasAllUnicastRoutes* sub-field of the *Options* parameter of the Proxy Table entry for this GPD.
For groupcast communication, the GPP 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, the GPP use 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 GPP 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 55 and Figure 56.

GPP2

Figure 55 – Exemplary message sequence chart for Green Power unicast communication

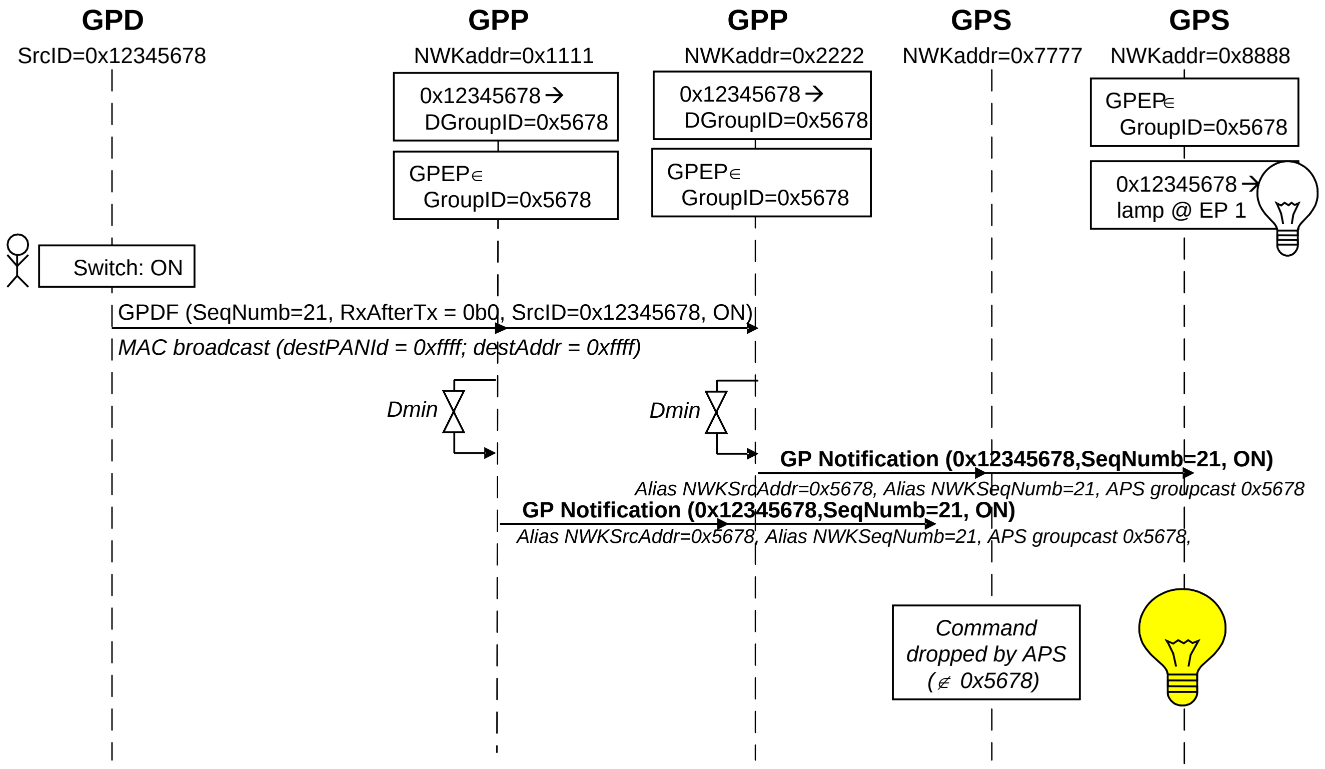

GPD
SrcID=0x12345678
GPP
NWKaddr=0x1111
0x12345678→ DGroupID=0x5678
GPEP∈ GroupID=0x5678
GPP
NWKaddr=0x2222
0x12345678→ DGroupID=0x5678
GPEP∈ GroupID=0x5678
GPS
NWKaddr=0x7777
GPS
NWKaddr=0x8888
GPEP∈ GroupID=0x5678
0x12345678→ lamp @ EP 1
Switch: ON
GPDF (SeqNumb=21, RxAfterTx = 0b0, SrcID=0x12345678, ON)
MAC broadcast (destPANId = 0xffff; destAddr = 0xffff)
Dmin
Dmin
GP Notification (0x12345678,SeqNumb=21, ON)
Alias NWKSrcAddr=0x5678, Alias NWKSeqNumb=21, APS groupcast 0x5678
GP Notification (0x12345678,SeqNumb=21, ON)
Alias NWKSrcAddr=0x5678, Alias NWKSeqNumb=21, APS groupcast 0x5678,
Command dropped by APS (∉ 0x5678)

Figure 56 – Exemplary message sequence chart for GP groupcast communication when RxAfterTx = 0b0

The proxy behaviour 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.3.8, 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 GPP can passively discover the information by storing pairing information from GP Notification and GP Tunnelling 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 neighbour network or on GPDs removed from the network.

A.3.5.2.2.1 Proxy Table entry status

The GPP 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 44.

Table 44 – Proxy Table entry status

EntryActive	EntryValid	Meaning
1	1	(According to this GPP's knowledge) The GPD with this GPD ID belongs to this ZigBee network, the sink information is current and valid.
1	0	(According to this GPP's knowledge) The GPD with this GPD ID 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 GPP's knowledge) The GPD with this GPD ID 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 GPP's knowledge) The GPD with this GPD ID 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, Sequence number, SearchCounter).

A.3.5.2.2.2 Maintenance

The GPP ***shall*** persistently store the pairing information, incl. the security settings for the paired GPSs, across restarts. On restart, the GPP ***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 GPP ***should*** rediscover its inactive Proxy Table entries. The GPP 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.

On receipt of GP Pairing command, the GPP ***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 GPS or CT sending the GP Pairing command.
If the GPP has no Proxy Table entry for this GPD, 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 GPP ***shall*** set its Proxy Table entry for this GPD to inactive and valid; all GPS flags ***shall*** be cleared and all GPSs removed.
If the Proxy Table entry becomes empty, i.e. if its *Sink address list* contains an address of a single GPS, and the GPP 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 GPP 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 GPP 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 GPP 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 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 GPP 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 GPEP as a member of this group in its *apsGroupTable*.

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 GPP 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). The proxy ***may*** also derive the DGroupID and add its GPEP 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 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 GPSs 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 GPP ***should*** send GP Pairing Search command with the *Request GPD Security Frame Counter* set if the *SecurityLevel* was 0b01, and/or *Request GPD Key*, 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 exists, the GPP 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 GPP 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 communication modes, then it stops the gppDiscoveryDelay timer and performs Active discovery.
- a GP Tunneling Stop command for this GPD ID,; 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, 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, 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, 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[2], and performs Active discovery.
- ▪ inactive and the SearchCounter equals 0, the GPP performs Active re-discovery.
- ▪ inactive and the SearchCounter differs from 0, the GPP 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 GPP initiates a timer with *gppDiscoveryDuration*. If at least one GP Pairing command is received within *gppDiscoveryDuration*, the Proxy Table entry for this GPD is marked as active and valid, and data from each such GP Pairing command is stored. Otherwise, if at *gppDiscoveryDuration* the Proxy Table entry for this GPD does not include any sink address, group or individual, the Proxy Table entry for this GPD is marked as inactive and invalid, and the Search counter is incremented by 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 Sinks flag* in the GP Pairing Search command; the other two sink request flags 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 flag*. The *Request Commissioned groupcast destinations* 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 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 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).

[2] 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: GPS 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 GPP may read out interesting Proxy Table entries of other GPP, 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 GPP 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 Proxy minimum operation

On receipt of GP-SEC.request, the Proxy Minimum acts as described in sec. A.3.7.2.1.

On receipt of GP Pairing command, the GPPm updates its Proxy Table. If the *RemoveGPD* flag was set to 0b1, the GPPm removes the Proxy Table entry for that GPD. If the *RemoveGPD* flag was set to 0b0; and the *AddSink* flag was set to 0b0, the GPPm removes the Sink group address from the *SinkGroupList*. If the *RemoveGPD* flag was set to 0b0 and the *AddSink* flag was set to 0b1, the GPPm adds the communication mode, if new, and the GPS (group) address, if not already included in the corresponding *SinkList*, and sets the entry to active and valid. The Proxy Minimum updates the Proxy Table fields *SecurityLevel*, *KeyType*, *GPDkey* and *GPDsecurityFrameCounter*, if they were included in the GP Pairing command. If the *Assigned Alias* field is present, the Proxy Minimum stores it in its Proxy table entry, and sets the corresponding *Options* sub-field. If the GPPm receives GP Pairing command with a *CommunicationMode* sub-field set to a value it does not support, it ***shall*** silently drop the packet.

On receipt of GP-DATA.indication, the Proxy Minimum checks the GPDF type. If the GPDF carries a GPD Commissioning command or a GPD Decommissioning command or a GPD Success command, the packet is silently dropped.

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 Proxy Minimum searches its Proxy Table for a matching entry related to the received GPD ID. If there is a Proxy Table entry for this GPD with the *InRange* flag set to 0b0 (even if the *GPDfixed* flag is also set to 0b1), the Proxy Minimum sets the *InRange* flag to 0b1. Then, the Proxy Minimum continues as follows.

The Proxy Minimum checks the security level of the received GPDF as follows. The GPPm 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 GPPm 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 Minimum stops processing the frame. The GPPm ***shall not*** send GP Notification or GP Pairing Search.

If all the checks succeed, the GPPm stores the *Sequence Number / Frame Counter* in the Proxy Table entry, and constructs from the received GPDF a GP Notification command(s) for each group address stored in the Proxy Table for this GPD. 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 GPPm 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.

For groupcast GP Notification, the GPPm shall further use the following values: NWK Src address and NWK Sequence Number: proxy alias (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: GPEP.

A.3.5.2.4 GPCm operation

On receipt of GP Pairing Configuration command, the GPCm shall act as GPS (see A.3.5.2.5).

On receipt of GP-SEC.request, the Combo Minimum acts as described in sec. A.3.7.2.1.

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 GPCm performs duplicate filtering, as described in A.3.6.1.2. Unicast GP Notifications for a GPD with groupcast Sink Table entry are silently ignored. Then the GPCm checks if it has a Sink Table entry for this GPD. If the GPCm does not have a Sink Table entry for this GPD, the command is silently ignored.
If the GPCm has a Sink Table entry for this GPD, 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 GPCm 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 GPCm updates the *Sequence number* or *Security Frame counter* field of the Sink Table entry.

If the GPCm supports bidirectional communication, it checks if the received GPD command does require response. If the received GPD command requires response and GPCm has a Translation Table, the GPCm checks if there is a Translation Table entry with value of the *EndPoint* field other than 0x00 and 0xfd. If no corresponding entry is found, no response is sent. If yes, the GPCm checks if the GPD has *RxOnCapability*. This information is available in the GPD command, if received directly, or from the Sink Table entry, if received in GP Notification. If not, the no response is sent. If yes, GPCm selects TempMaster as described in sec. A.3.6.2.3. If GPCm itself is selected as TempMaster, the GPS calls GP-DATA.request, with the required *GPD CommandID* and *GPD Command Payload*. Then, and also if the received GPD command does not require response, or the GPCm does not support bidirectional communication, the GPCm acts as follows.
If the GPD command was received directly in GP-DATA.ind, the GPCm constructs and sends a GP Notification command, taking the parameters from the Sink Table: *CommunicationMode* subfield of the *Options* field; *Pre-Commissioned Groupcast* 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.

Then and if the GPD command was received in GP Notification, and the GPCm has a Translation Table, the GPCm 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 GPCm 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.

On receipt of GPD commissioning commands, the GPCm acts exactly as described for GPS (see A.3.5.2.5).

A.3.5.2.5 GPS operation

A GPS ***should*** re-announce its pairings when it rejoins the network (e.g. after being powered off) by sending a GP Pairing command.

On receipt of ZigBee Update Device and Device_annce commands with IEEE address other than 0xffffffffffffffff, the GPS 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 in A.3.6.3.4, using the conflicting Alias NWK source address and a fixed sequence number of 0x00, to force the regular ZigBee device to change its short address. The alias ***shall not*** be changed.

On receipt of GP-SEC.request, the GPS acts as described in sec. A.3.7.2.1.

On receipt of a GP Commissioning Notification with *Security processing failed* sub-field of the *Options* field set to 0b0, the GPS 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, GPS checks if it is in commissioning mode. If not, the GP Commissioning Notification command, Decommissioning GPDF and Commissioning GPDF is silently dropped.

On receipt of GPD Decommissioning command in commissioning mode, the GPS checks if it has a Sink Table entry for this GPD. If not, the frame is ignored. If yes, the GPS 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 GPS removes this Sink Table entry, removes/replaces with default entries the corresponding Translation Table entries if Translation Table functionality is supported, and removes GPEP membership at APS level in the groups listed in the removed entry, if any. Then, the GPS schedules sending of a GP Pairing command for this GPD, with the *RemoveGPD* sub-field set. If the removed Sink Table entry included any pre-commissioned groups, the GPS ***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 GPS supports Single-hop 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 GPS 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 GPS silently drops, irrespective of whether it is in operational mode or in commissioning mode.

The GPS reaction on reception of GP Pairing Configuration is the same, irrespective of whether it is in commissioning mode or operational mode.

On receipt of GP Pairing Configuration command, the GPS 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.

If the *Action* sub-field of the *Actions* field is set to 0b000, the GPS ***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 in the Sink Table, the GPS ***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 in the Sink Table, the GPS ***shall not*** send the GP Pairing command(s).

For *Action* sub-field equal to 0b001 or 0b010, the GPS starts as follows. The GPS checks if it supports the *SecurityLevel* requested (i.e., if it is higher than 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. If both checks succeed, the GPS proceeds as follows, depending on the *Action* sub-field value.

If the *Action* sub-field of the *Actions* field is set to 0b010, the GPS ***shall*** remove all the Sink Table entry/entries for this GPD, if any. For all the removed groupcast pairings, the GPS ***shall*** remove its GPEP as a member of the group at APS level. If the GPS has any Translation Table entry/entries for this specific GPD ID, they all ***shall*** be removed or replaced with the default Translation Table entry.

Both for *Action* sub-field equal to 0b001 if there is no Sink Table entry for this GPD ID and 0b010, the GPS ***shall*** then analyze the *Number of paired endpoints* field.
If the *Number of paired endpoints* field is set to 0x00 or 0xfd, there data from this GPD is not meant for local execution on this GPS. If the GPS does support *SinkTable-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, with the *EndPoint* field having the value 0xfd. If the *CommunicationMode* supplied in the Pairing Configuration command was groupcast, the GPS ***shall*** add its GPEP as a member of the supplied group or derived group at APS level if not already a member. If the GPS does NOT support *SinkTable-based forwarding* or it does not support *SinkTable-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 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 (GPS ***shall not*** use this option if it has default 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. If the Sink Table entry is created and the *CommunicationMode* supplied in the Pairing Configuration command was groupcast, the GPS ***shall*** add its GPEP 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 GPS ***may*** then create a Sink Table entry with the supplied information and Translation Table entry for the GPD ID, with the *EndPoint* field having the value 0xff; the unmodified default entry, if available, ***may*** be used instead. If the *CommunicationMode* supplied in the Pairing Configuration command was groupcast, the GPS ***shall*** add its GPEP 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 GPS. 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, GPS is to derive the paired endpoints in an application-specific manner. The GPS ***should*** then create a Sink Table entry with the supplied information and a Translation Table entry/entries for the GPD ID, with the *EndPoint* field containing the derived value; the unmodified default entry, if available, ***may*** be used instead. If the *CommunicationMode* supplied in the Pairing Configuration command was groupcast, the GPS ***shall*** add its GPEP 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 has values other than 0x00, 0xfd, 0xfe, or 0xff, the *Paired endpoints* field is present and contains the list of local endpoints paired to this GPD; the GPS creates a Translation Table entry for this GPD ID and each EndPoint. If the *CommunicationMode* supplied in the Pairing Configuration command was groupcast, the GPS ***shall*** add its GPEP 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 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. If the *CommunicationMode* does match, the sink checks the *Number of paired endpoints* field. If set to 0xff, 0xfe or value other than 0x00, 0xfd, 0xfe, or 0xff; 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 GPS ***shall*** add its GPEP 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 0b011, the GPS ***shall*** check if it has Sink Table entry for the supplied *SrcID/GPD IEEE address* 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 GPS ***shall*** remove its GPEP as a member of this group at APS level. If the GPS has any Translation Table entry/entries for this GPD ID and endpoint, if specific endpoint is provided in the GP Pairing Configuration command, they ***shall*** be removed/replaced with the default Translation Table entry.

If the *Action* sub-field of the *Actions* field is set to 0b100, the GPS ***shall*** remove all the Sink Table entry(s) for this GPD, if they exist. For all the pairings that were for groupcast, the GPS ***shall*** remove its GPEP as a member of the group at APS level. If the GPS has any Translation Table entry/entries for this GPD ID, they all ***shall*** be removed/replaced with the default Translation Table entry.

If the *Send GP Pairing* sub-field of the *Actions* field of the GP Pairing Configuration command is set to 0b1, the GPS ***shall***, upon completion of Sink Table update, send the GP Pairing command(s) reflecting the changes made and, if a pairing was added, it ***shall*** send a Device_annce command for the alias. If the *Send GP Pairing* sub-field of the *Actions* field was set to 0b0, the GPS ***shall not*** send the GP Pairing command or Device_annce command.

If the sink implements the Proxy table maintenance functionality, the sink ***shall*** act as follows. The GPS 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 GPS checks if it has a Sink Table entry for this GPD 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 GPS 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 GPS checks if it has a Sink Table entry for this GPD. If the *SecurityLevel* and *SecurityKeyType* check, freshness check and security processing all pass successfully, the GPS 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.

On reception of GP-DATA.indication with Status AUTH_FAILURE, the GPS 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 GPS has GP stub implemented (GPT+ or GPC), the GPS performs duplicate filtering, as described in A.3.6.1.2. Then the GPS checks if it has a Sink Table entry for this GPD. If not, and the GPD command was received in unicast GP Notification, it schedules sending of GP Notification Response, if supported, in unicast to the originating proxy, with the *No Pairing* flag set to 0b1, as well as broadcasting of a GP Pairing command with the *CommunicationMode* flag set to 0b00 and *AddSink* flag set to 0b0. If the GPS does not have a Sink Table entry for this GPD, and the GPD command was received directly or in groupcast, the command is silently ignored. If the GPS has a Sink Table entry for this GPD for groupcast communication mode (0b01 or 0b10) and it receives unicast GP Notification, the GPS 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 the *CommunicationMode* flag set to 0b00 and *AddSink* flag set to 0b0.
If the GPS does have a Sink Table entry for this GPD, 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 GPS 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 GPS updates the *Sequence number* or *Security Frame counter* field of the Sink Table entry, if present, and proceeds as follows.

If the GPS 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 indicates any groupcast *CommunicationMode*, and there is no Translation Table (if supported) entry for this GPD ID and GPD CommandID with *endpoint* field set to 0x00, the GPS ***shall*** construct and send a GP Notification command for each of the paired groups, taking the following parameters from the Sink Table: *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.

Then, the sink checks if the command requires response. If the received GPD command does not require response, the GPS executes the command. If the received GPD command requires response, the GPS checks if the GPD requesting it has *RxOnCapability*. This information is available in the GPD command, if received directly, or from the Sink Table entry, if received in GP Notification. If not, the command is silently dropped. If yes, GPS selects TempMaster as described in sec. A.3.6.2.3. If GPS is selected as TempMaster, the GPS calls GP-DATA.request, with the required *GPD CommandID* and *GPD Command Payload*.

The sink behaviour 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 *Security processing failed* 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.3.8, 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 GPP 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 GreenPower 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 GPSs listed in its Proxy Table for this GPD (if any), as specified in section A.3.5.2.1.

A.3.6 GP Implementation details

A.3.6.1 Generic

This chapter describes functionality common to all GreenPower cluster implementations, both on GPPs and GPSs.

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 GreenPower cluster messages, the destination endpoint ***shall*** be set to 0xf2.

A.3.6.1.2 Duplicate filtering

In the GPEP duplicate filter, each entry is stored for a finite time of *gpDuplicateTimeout* and is used to filter both direct and tunnelled 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* 0b01, 0b10 or 0b11, then the filtering of duplicate messages is performed based on the *GPD security frame counter*.

If the receiving device is:

- a GPP,
- a GPS and it does not support bidirectional communication,
- a GPS does support the bidirectional communication but the RxAfterTx/AppointTempMaster 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 GPS, it does support the bidirectional communication and the *RxAfterTx /AppointTempMaster* is set to 0b1, then the GPS processes further - independent of the manner of receiving the GPD command: directly as GPDF or indirectly in a GP command - each instance of this command, with *GPP distance* shorter than the last received one, or by the same *GPP distance* – with the lower short address. The distance and the address shall then be also stored.

In case of duplicate unicast GP Notification, the GPS ***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 45 summarizes the duplicate filtering in the GPS GPEP, dependent on the required and received *CommunicationMode* and the *RxAfterTx* value.

Table 45 – Duplicate filtering in GPS

Required communication mode	**Communication mode of first packet**	**RxAfterTx (Appoint TempMaster)**	**Action**
Derived group	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 and *AddSink* flag set to 0b0.
Pre-commissioned group	Unicast	TRUE/FALSE	
Unicast, Pre-commissioned group	Derived group	TRUE/FALSE	drop packet, don't store the new values in the duplicate filter
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

Required communication mode	Communication mode of first packet	RxAfterTx (Appoint TempMaster)	Action
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
Unicast	Unicast	FALSE	For the first received 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 shorter distance (or same distance, lower address), store the new values in the duplicate filter
Pre-commissioned group	Pre-commissioned group	TRUE	
Any	GPDF (direct mode)	TRUE	pass packet up if shorter distance (or same distance, 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 shorter distance (or same distance, lower address), store the new values in the duplicate filter, send GP Pairing with the proper communication mode; can be modified by the profile
Unicast	Unicast	TRUE	For the first received unicast packet: Send GP Notification Response with *FirstToForward* sub-field of the *Options* field set to 0b1, pass packet up if shorter distance (or same distance, lower address), 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), pass packet up if shorter distance (or same distance, lower address)

A.3.6.1.2.1 gpDuplicateTimeout

The time the GPEP of the GPS and GPP keeps the information on the received GPDF with random sequence number, 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, the filtering of reply GPD commands depends of the *MAC sequence number capabilities* of a particular GPD. For random sequence numbers, any number that passes the duplicate filter is accepted. For incremental numbers, the received MAC sequence number must be higher than the last sequence number stored in the Proxy/Sink table, taking into account counter roll over, see A.3.6.1.3.1.

If the GPD command used *SecurityLevel* 0b01, 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. 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.3.6.1.3.1 MAC sequence number rolling over

For the incremental sequence number (when *SecurityLevel* = 0b00), the counter must be allowed to roll over, because of the limited sequence number length of 1 octet, so care must be taken when comparing for freshness.

It is recommended that this comparison be accomplished as follows:

```
define
a = sequence number stored by the GPP/GPS;
b = sequence number from the GPDF;
if 1 ≤ ( (b – a)mod 256 ) < 128        accept GPDF;
else                                   drop GPDF;
```

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 tunnelled 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 proxy alias (see A.3.6.3.3).

A.3.6.1.5 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 57), reading out GPD attribute (Figure 58) and GPD requesting an attribute (Figure 59).

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 Attribute 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.

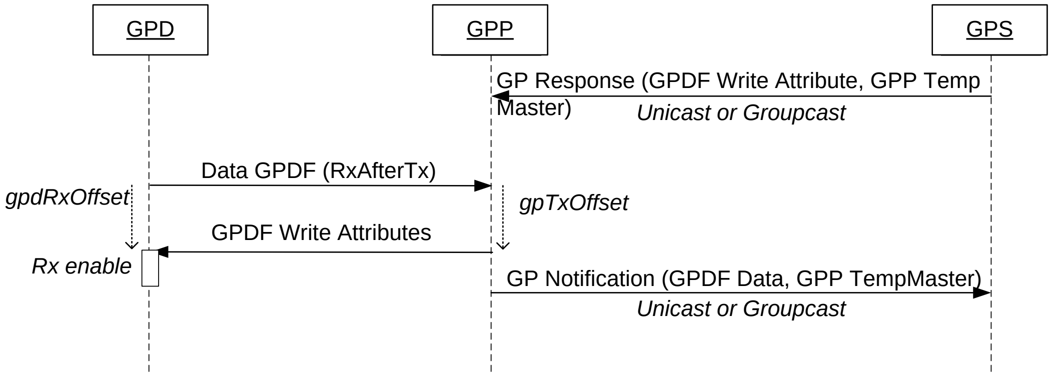


Figure 57 – MSC for GP bidirectional operation: writing into GPD

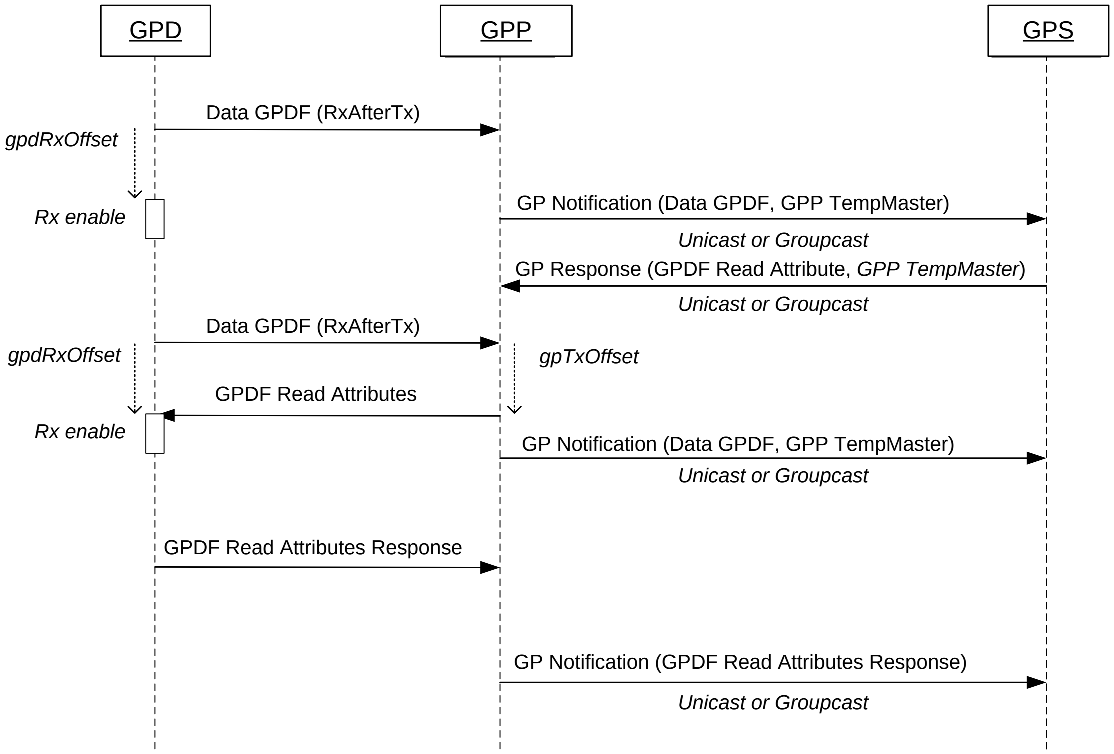

GPD
GPP
GPS
Data GPDF (RxAfterTx)
gpdRxOffset
Rx enable
GP Notification (Data GPDF, GPP TempMaster)
Unicast or Groupcast
GP Response (GPDF Read Attribute, GPP TempMaster)
Unicast or Groupcast
Data GPDF (RxAfterTx)
gpdRxOffset
gpTxOffset
GPDF Read Attributes
Rx enable
GP Notification (Data GPDF, GPP TempMaster)
Unicast or Groupcast
GPDF Read Attributes Response
GP Notification (GPDF Read Attributes Response)
Unicast or Groupcast

Figure 58 – MSC for GP bidirectional operation: reading out GPD attribute

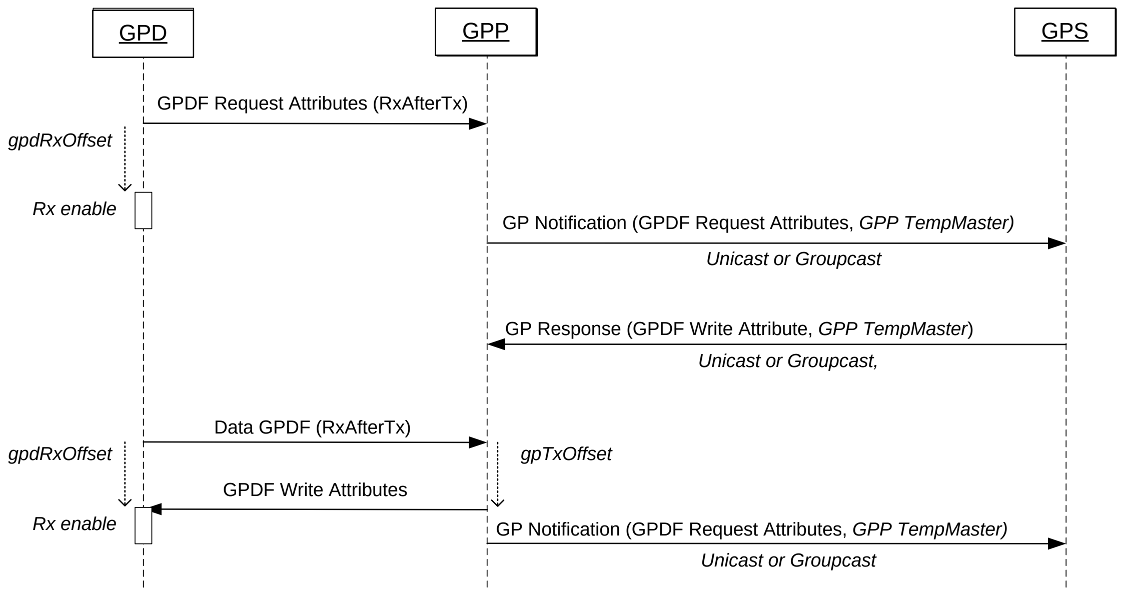

GPD
GPP
GPS
GPDF Request Attributes (RxAfterTx)
gpdRxOffset
Rx enable
GP Notification (GPDF Request Attributes, GPP TempMaster)
Unicast or Groupcast
GP Response (GPDF Write Attribute, GPP TempMaster)
Unicast or Groupcast,
Data GPDF (RxAfterTx)
gpdRxOffset
gpTxOffset
GPDF Write Attributes
Rx enable
GP Notification (GPDF Request Attributes, GPP TempMaster)
Unicast or Groupcast

Figure 59 – 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.

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 GPS needs to translate GPD specific application functionality (GPDF device identifiers and GPD commands) relevant for GPS's application endpoints into ZigBee ZCL commands. One way to solve it is to implement the Translation Table, as defined below.

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 GPS contains a *GPD Command Translation Table,* each entry of which is formatted as shown in Table 46.

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 46 – 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 /0xffffffff ffffffff	Identifier of the GPD
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.

The *Options* field shall be formatted as shown in Figure 60.

Bits: 0..2	3..7
ApplicationID	Reserved

Figure 60 – 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. *ApplicationID* = 0b010 indicates the *GPD_ID* field has the length of 8B and contains the GPD IEEE address. All values of *ApplicationID* other than 0b000 and 0b010 are reserved in the current version of the GreenPower cluster specification.

The *ZigBee Command payload* field is formatted as defined in Figure 61.

Octets	1	Variable
Data Type	unsigned 8-bit integer	set of unsigned 8-bit integer
Field Name	Length	Payload

Figure 61 – Format of the ZigBee Command Payload field of the Translation Table entry

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 *GPD Command* field is set to 0xAF, all of the GPD sensor report commands 0xA0 – 0xA3 are supported.

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 used. In 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.

There ***should*** be only one entry in the GPD Command Translation Table for each (GPD ID, GPD Command, EndPoint, ZigBee Profile, ZigBee Cluster) quintuple.

Note that for a single GPD ID, there may be multiple entries, for multiple GPD commands.

Note that for a single GPD ID the same GPD Command could result in different translated ZigBee CommandIDs, for different EndPoint, Profile and Cluster values.

By default, the GPD Command Translation Table ***may*** contain the default translations (see Table 48 and Table 49) for all GP-controllable application functionality, for *ApplicationID* = 0b000 and *SrcID* 0xffffffff and/or, if supported, for *ApplicationID* 0b010 *GPD IEEE address* 0xffffffffffffffff. If no default generic translations are available, default Translation Table entries ***shall*** be added upon successful completion of single-hop and multi-hop commissioning, and upon reception of GP Pairing Configuration leasing to Sink Table entry creation (as described in A.3.5.2.5).

The default GPD Command Translation Table entry can be overwritten with the GP Translation Table Update command.

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 GPS creates a list of candidate responders, consisting of the GPPs which did forward GP Notification command, if any, as well as itself, if it did receive the GPD command directly. The GPS selects the node with the shortest distance to this GPD, or if multiple have the same distance, the one with shortest distance and lowest short address.

If another device is chosen as the TempMaster, the GPS sends the GP Response frame carrying the APPL data payload (*GPC 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 GPS itself is chosen as the TempMaster, it ***should*** broadcast the GP Response, and it ***shall*** then carry the short address of the GPS in the *TempMaster short address* of the payload.

A.3.6.3 Proxy implementation

A.3.6.3.1 gppTunnelingDelay

The gppTunnelingDelay is calculated, taking into account the following criteria:

- distance to the GPD, as indicated by the LQI;

- Fact of being first to forward for the previous GPDF from this GPD;
- If unicast communication mode: knowledge of the route to the GP sink and path cost to the sink.

The gppTunnelingDelay can be calculated according to the following formula

$$\text{gppTunnelingDelay [ms]} = \begin{cases} \text{Dmin;} & \text{if FirstToForward = TRUE \& NoRoute=FALSE} \\ 85 \cdot (1- ((\text{Dmax} - d)/\text{Dmax})\text{\^{}}6) + \text{Dmin;} & \text{if FirstToForward = FALSE \& NoRoute=FALSE} \\ \text{Dmin + Dmax; if NoRoute=TRUE} & \end{cases}$$

where:

- Dmin = 15 ms
- Dmax=100ms
- d is the distance to the GPD, as indicated by the LQI;
- NoRoute is a Boolean flag: as stored in the Proxy Table entry for this GPD.
- FirstToForward is a Boolean flag, as stored in the Proxy Table entry for this GPD.

Note, that for any communication mode, the ZigBee stack adds additional randomized delays.

A.3.6.3.1.1 LQI – distance relationship

Each stack vendor is responsible for producing normalized LQI values, corresponding to line-of-sight distance to GPD sending at 0dBm.
In order to provide such normalized LQI values, a test system shall be constructed by each stack vendor, which consists of two nodes: one node configured to transmit GPD frames at nominal 0dBm transmit power, and a second node to receive the frames and report the LQI that is calculated by the MAC. The nodes shall be placed apart in free space with no nearby obstructions or surfaces that would reflect or disrupt the RF signal. Measurements of the LQI at the second node shall be taken when the devices are separated by various fixed distances (e.g. 1m, 2m, 5m, 10m). It is recommended that the average of a series of measurements is taken at each distance. The LQI values obtained, suitably interpolated and extrapolated as required, shall then be used to map the full range of LQI values (0 to 255) to an approximation of the distance between devices.

These distance values will then be used by the algorithm that is used for *gppTunnelingDelay* calculation.

A.3.6.3.1.2 Calculation

The function for distance-based *gppTunnelingDelay* calculation (see sec. A.3.6.3.1), if too complex to be efficiently implemented, ***should*** be approximated by the following table.

Table 47 – Approximation of gppTunnelingDelay

Distance range [m]	gppTunnelingDelay [ms]
0	15
1	20
2	25
3	29
4	33
5	38

Distance range [m]	gppTunnelingDelay [ms]
6	41
7	45
8	48
9	52
10	55
11	58
12	61
13	63
14	66
15	68
16	70
17	72
18	74
19	76
20	78
21	79
22	81
23	82
24	84
25	85
26	86
27	87
28	88
29	89
30	90
31	91
32–33	92
34	93
35-36	94
37-38	95
39-41	96
42-44	97
45-48	98
49-57	99
58-100	100

A.3.6.3.2 gppCommissioningWindow

The default value is to be defined by the application profile endorsing the Green Power feature. The default value for the GPP, *gppCommissioningWindow*, can be overwritten by the GPS 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 GPS is capable of filtering the GP Notification commands at the GPEP 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 GPSs, to filter the messages sent by the proxies on behalf of the same GPD, the proxies originating the message use 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 GPEP will be able to filter out, based on the full GPD ID 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.

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: 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. 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 which – both for assigned and derived alias - has the 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 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* – 13 (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 single-hop or multi-hop commissioning.

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 as NWK source address and a fixed NWK sequence number 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 62.

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 GPP	0

Figure 62 – 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 Security assumptions

Four security levels for GPDF frame protection are offered by the specification, as summarized in A.1.5.3.2. 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 GPSs can only be controlled by GPDs with matching (security) capabilities.

Two-step security processing of the incoming GPDF is performed: GPPs authenticate and check the freshness of the frame, before forwarding; and GPS check the required security level and frame freshness before execution.

All GPP and GPS 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 perform an ACL and freshness check. This level only protects the system on runtime against genuine non-malicious devices which were not paired to this network, e.g. neighbour’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 GPS vendors.

The *SecurityLevel* 0b01 provides medium authentication and replay protection for the GPDF. It was the best GPDF authentication affordable by the of-the-shelf GPD equipped with electro-mechanical energy harvester at the time of this specification drafting. The GPD are expected to migrate towards the next higher security level when the technical advantages allow.

The packet length is minimized by sending only 1LSB of the frame counter and only 2LSB of the MIC are transmitted. Since 4B of the frame counter are used for the nonce, the receiver needs to reconstruct the full value by filling out the MSBs. Because of the distributed nature of the system and mobility of the GPDs, accepting other MSB values than the one recently received/recovered is possible, and controlled by the *gppSecurityWindow* parameter.
Both optimizations together influence the probability for a fake packet, with forged (random) MIC, created by an attacker, to pass authentication check. The higher the *gppSecurityWindow* value, the better the attacker's chances to forge a GPDF. For less critical applications, (like e.g. home lighting on/off switch), the value of *gppSecurityWindow* may be set to 0-1-2 (allowing the receiver to miss up to 254, 510 or 766 frames, respectively). For safety-critical applications (like e.g. industrial on/off switch) or security-critical applications (like e.g. door opener), the higher security levels are recommended.

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 (GPP, GPT+ or GPC) 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.3.7.2 Security operation

A.3.7.2.1 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 SecurityLevel = 0b01, it shall recover the full frame counter first, as described in A.3.7.2.4. 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 GPS (GPT+ or GPC) or a GPP.

If the device is a GPC, i.e. has both sink and proxy functionality, the Sink Table ***shall*** be consulted first. 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.3.7.2.2 GPS

The GPS (i.e. GPT+ and GPC) 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 GPS is a GPT+, it shall generate GP-SEC.response with the Status DROP_FRAME and the gppSecurityWindow 0.

If there no Sink Table entry for this GPD and the sink is in operational mode, and the GPS is a GPC, it shall act a described in A.3.7.2.3.

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 GPS fetches the shared key. If there is none, GPS generates GP-SEC.response, with the Status DROP_FRAME. If there is, the GPS generates GP-SEC.response, with the Status MATCH, and includes the key, the key type and the frame counter as processed here; the gppSecurityWindow shall be set to 0. 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 GPS generates GP-SEC.response, with the Status DROP_FRAME.

If there is a SinkTable entry for this GPD, 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. If any of those checks fails, the GPS generates GP-SEC.response, with the Status DROP_FRAME. If the checks are successful, the GPS generates GP-SEC.response, with the Status MATCH, and includes the key, the key type and the frame counter as processed here; the gppSecurityWindow shall be set to 0.

A.3.7.2.3 GPP

The GPP checks if it has a Proxy Table entry for this GPD.

If the proxy has an active entry, 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. 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 GPS generates GP-SEC.response, with the Status MATCH, and includes the key, the key type and the frame counter as processed here; the gppSecurityWindow is copied from the corresponding parameter.

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 GPS fetches the shared key. If the key type was 0b1 or the key type was 0b0 and there is no shared key, GPS generates GP-SEC.response, with the Status PASS_UNPROCESSED.

A.3.7.2.4 Incoming frames: frame counter recovery

On reception of GP-SEC.request for *SecurityLevel* = 0b01, the frame counter is obtained from the *GPDSecurityFrameCounter* field of Sink/Proxy Table for this GPD, and processed in the following way:

- If MAC *sequence number* field from the received GPDF has value greater than the LSB of the *incomingFrameCounter,* then the *FrameCounter* value for the GP-SEC.response is generated as follows: the LSB of the *incomingFrameCounter* is replaced with the value from the MAC *sequence number* and the resulting 4B number is used as frame counter for the Nonce.
- If MAC *sequence number* field from the received GPDF, as compared with the LSB of the *GPDSecurityFrameCounter* field of the Sink/Proxy Table entry, indicates that the LSB rolled over (see sec. A.3.6.1.3.1), then the *FrameCounter* value for the GP-SEC.response is generated as follows: then the LSB of the *incomingFrameCounter* is replaced with the value from the MAC *sequence number* and the remaining 3B part, if not 0xffffff, is incremented by one.

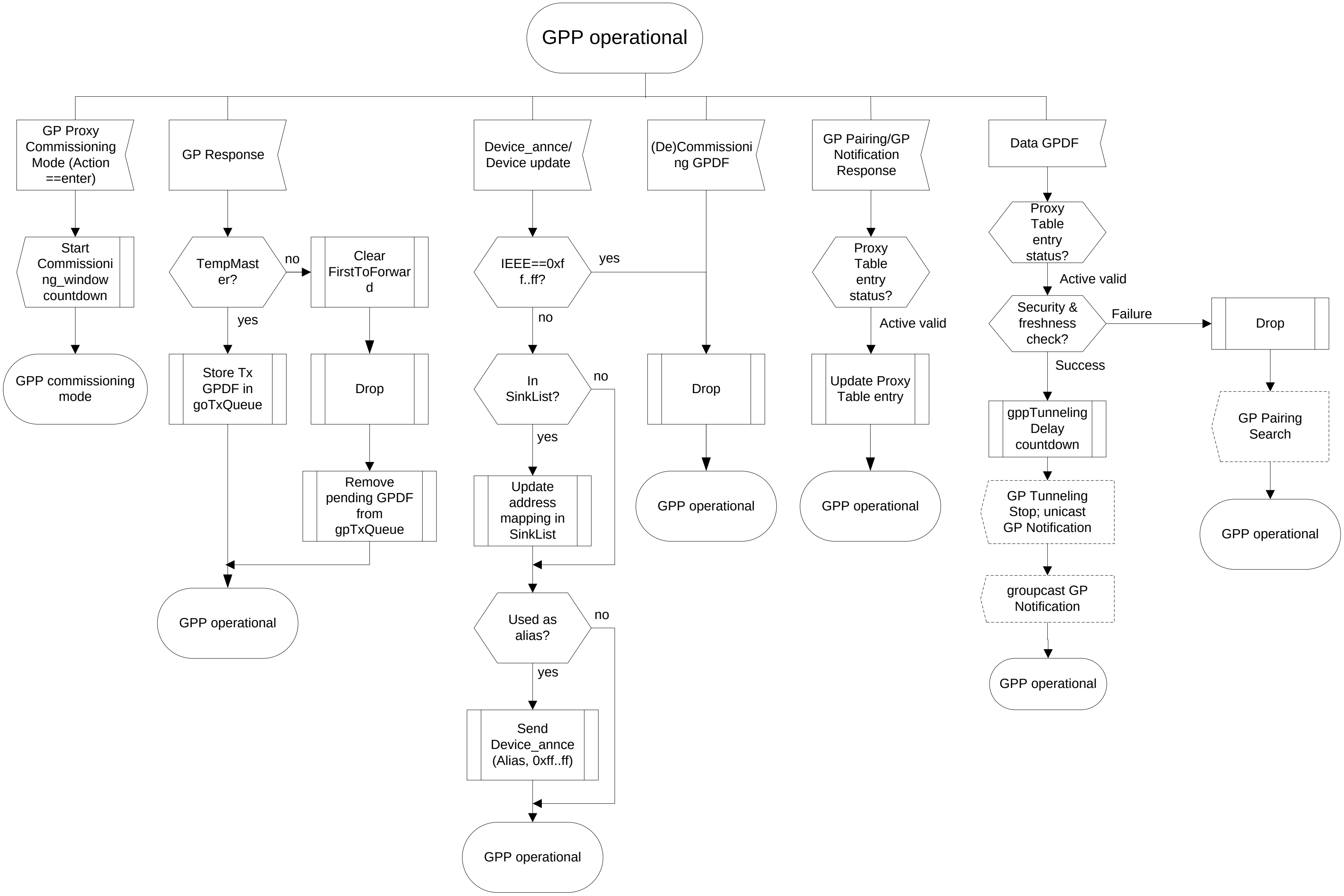
A.3.7.2.5 Incoming frames: key recovery

- 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,
    - 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.3.8 SDL diagrams for GreenPower cluster operation

In this section, SDL diagrams are included, to provide high-level overview of the GreenPower cluster operation. Please note, that this is high-level overview, and some detailed steps are not explicitly listed. Also, the application-specific behaviour is on purpose not included.


GPP operational
GP Proxy Commissioning Mode (Action ==enter)
GP Response
Device_annce/ Device update
(De)Commissioning GPDF
GP Pairing/GP Notification Response
Data GPDF
Start Commissioning_window countdown
GPP commissioning mode
TempMaster?
no
yes
Clear FirstToForward
Store Tx GPDF in goTxQueue
Drop
Remove pending GPDF from gpTxQueue
GPP operational
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
Drop
GPP operational
Proxy Table entry status?
Active valid
Update Proxy Table entry
GPP operational
Proxy Table entry status?
Active valid
Security & freshness check?
Failure
Drop
Success
gppTunneling Delay countdown
GP Tunneling Stop; unicast GP Notification
groupcast GP Notification
GPP operational
GP Pairing Search
GPP operational

1

2 Figure 63 – Proxy behavior in operational mode

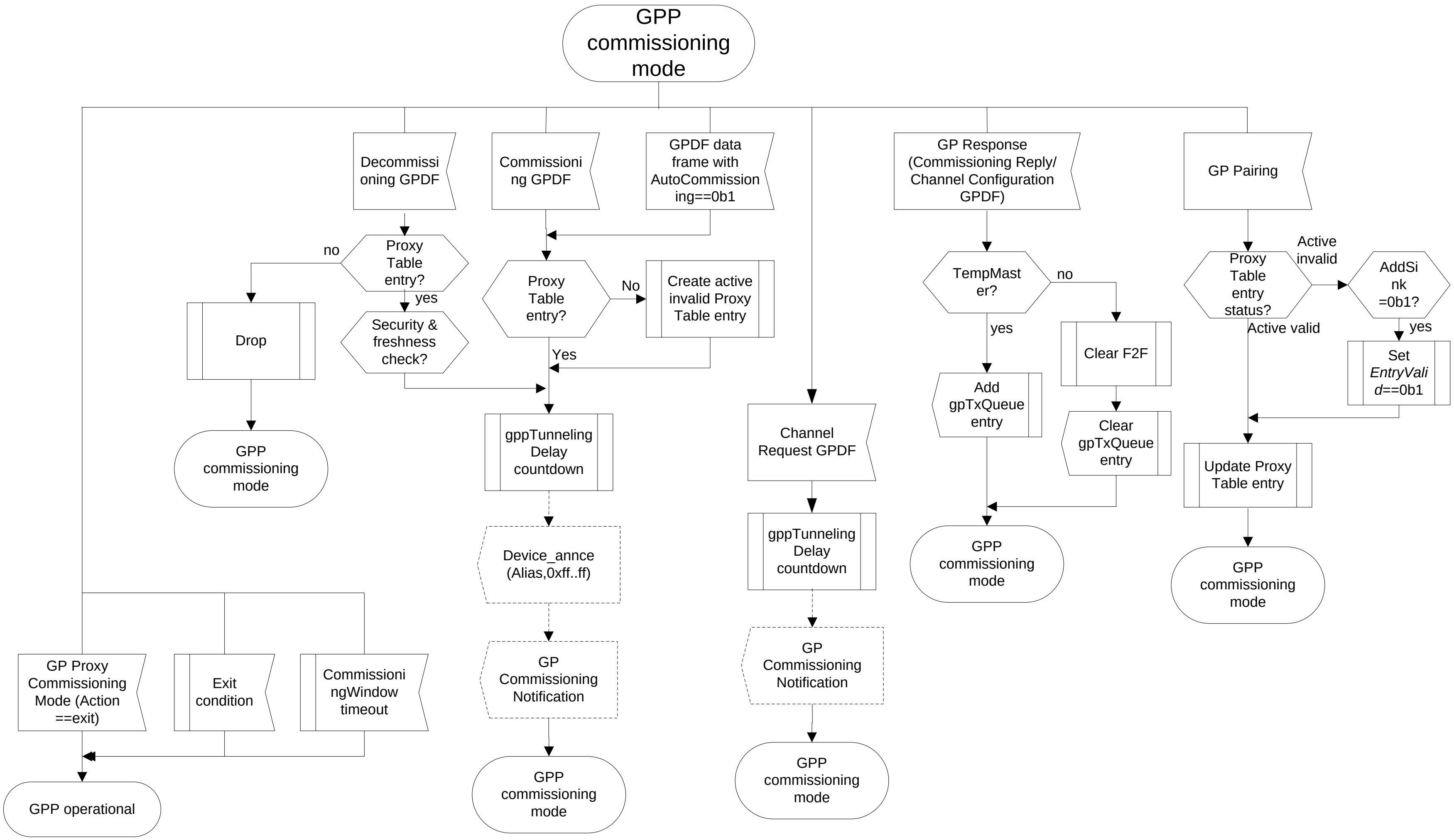
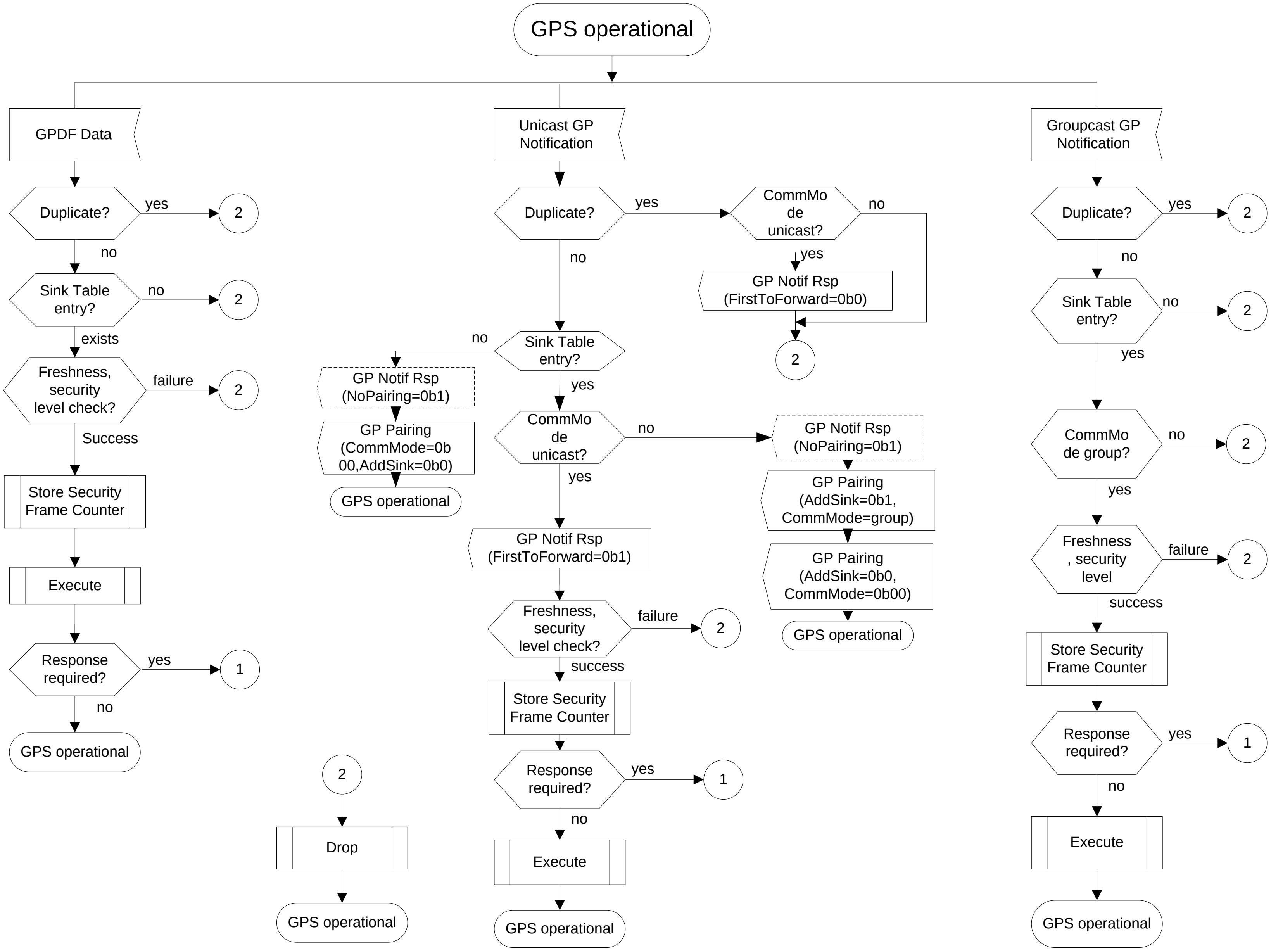


Figure 64 – Proxy behavior in commissioning mode

.



1

2 **Figure 65 – Sink behavior in operational mode (part 1)**

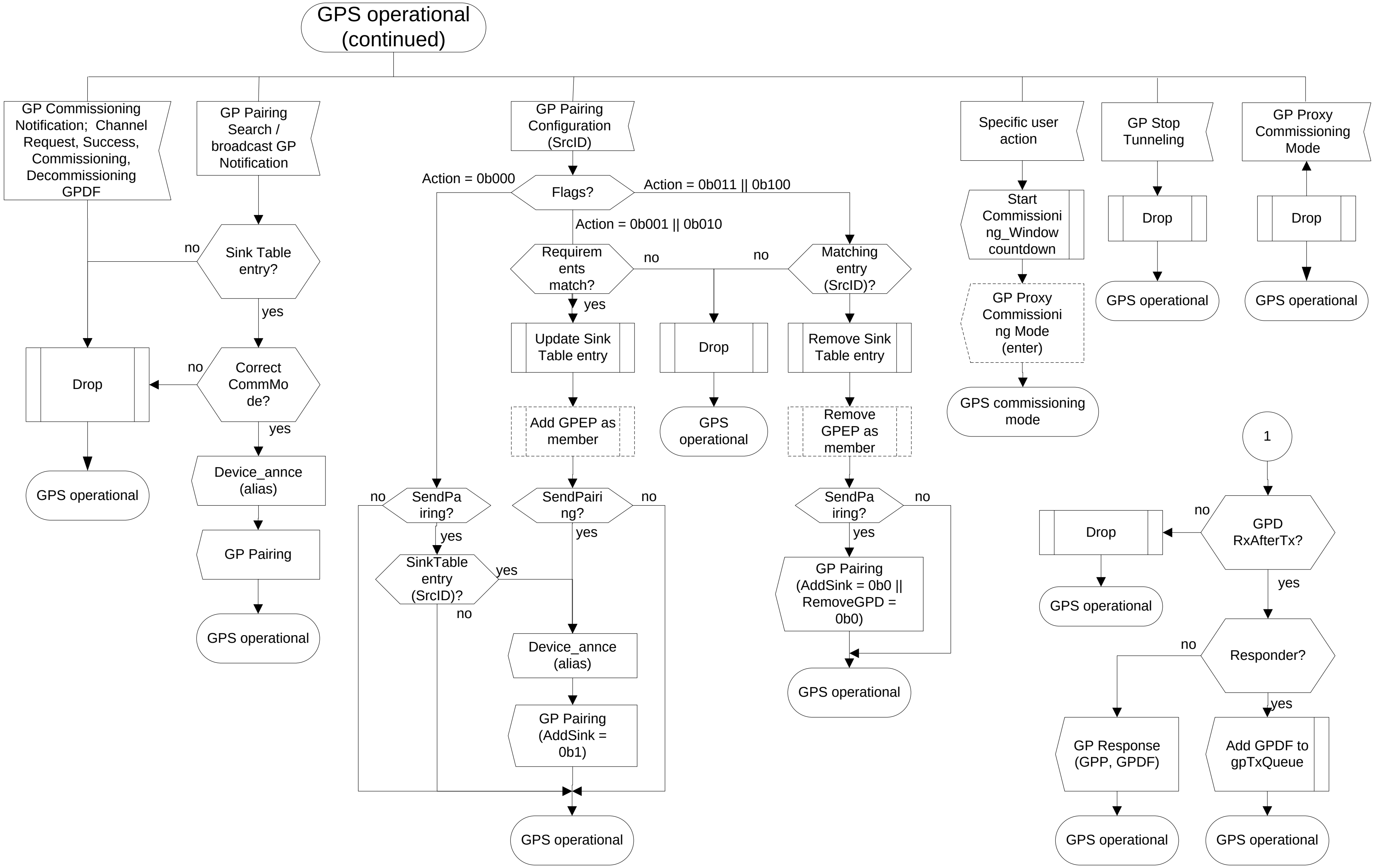


Figure 66 – Sink behavior in operational mode (part 2)

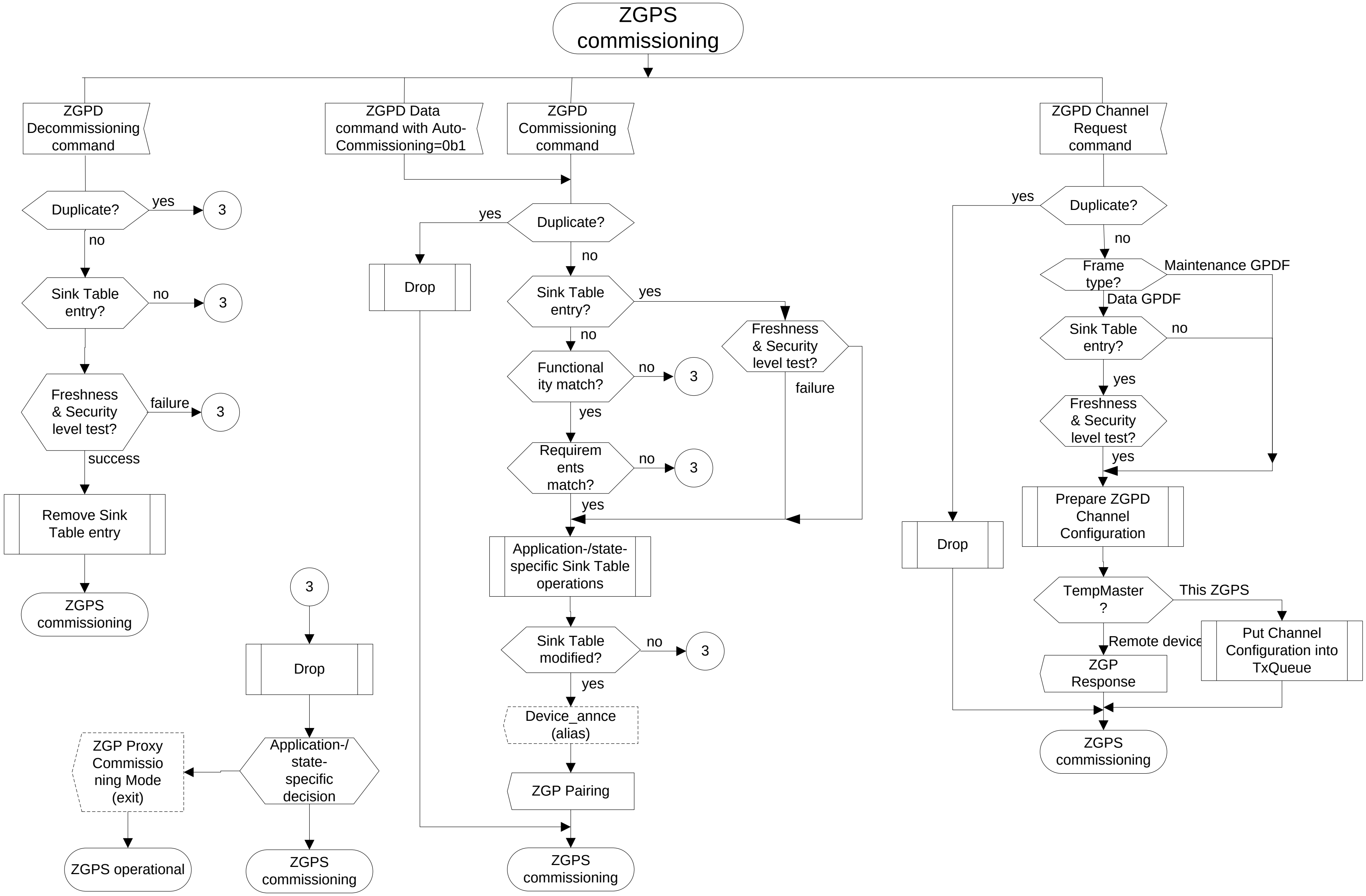


Figure 67 – Sink behavior in commissioning mode (part 1)

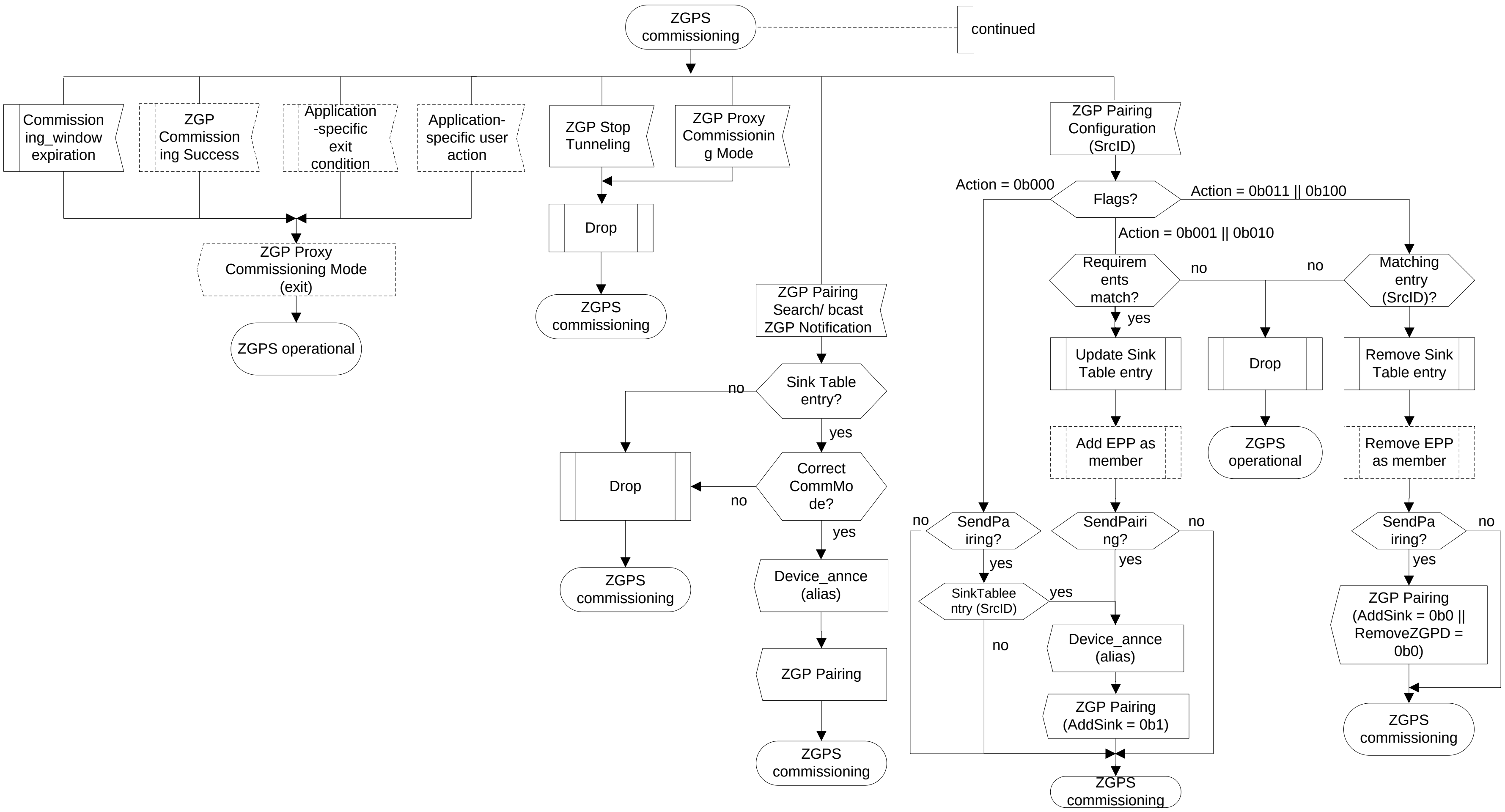


Figure 68 – Sink behavior in commissioning mode (part 2)

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 GreenPower cluster commands.

It is left to the implementers of GPS 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 GPS behaviour 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 GPS**: The user enables commissioning on the GPS via a vendor-specific action:
   a. The GPS enters commissioning mode
   b. Optionally (depending on the vendor-specific requirements) the GPS 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*, 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 proxy-based commissioning to indicate that this option is applied, and the term proximity-based commissioning to indicate that this option is not applied. In the proximity-based commissioning, the commissioned GPS and the GPD are the only involved parties.
2. **Proxies enter commissioning mode**: GPPs receiving a GP Proxy Commissioning Mode (*Action*=enter) command on the operational channel (if sent) in operational mode 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.
   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 repeats the commissioning action on the GPD **until success feedback or failure feedback is provided by the commissioning GPS**.
   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.
   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 *RxAfterTx*=0b1 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, with *RxAfterTx*=0b1; the Channel Request GPDF ***may*** be sent

protected.

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 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 proxy-based commissioning) or step 5 (for proximity-based commissioning)**.

b. If the GPD is in commissioning mode AND *RxAfterTx*=0b1 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 *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.

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.

iii. **GOTO step 4 (for proxy-based commissioning) or step 5 (for proximity-based commissioning)**.

c. If the GPD is in commissioning mode AND *RxAfterTx*=0b1 AND its internal *ToggleChannel* variable is FALSE AND its *ParametersStored* variable is TRUE, the GPD sends a Success GPDF on the operational channel; 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.

Note: If *gpdSecurityLevel* = 0b11, the Success GPDF ***shall*** be secured with either *SecurityLevel* = 0b10, if supported, or *SecurityLevel* = 0b11.

GOTO step 4 (for proxy-based commissioning) or step 5 (for proximity-based commissioning).

d. If the GPD is in commissioning mode AND *RxAfterTx*=0b0, and the GPD is capable of sending Commissioning GPDFs, the GPD sends a Commissioning GPDF on one channel, with *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. The GPD ***should*** start with the last memorized channel.

GOTO step 4 (for proxy-based commissioning) or step 5 (for proximity-based commissioning).

e. If the GPD is in commissioning mode AND *RxAfterTx*=0b0 and the GPD is not capable of sending Commissioning GPDF, i.e. Data GPDF with Auto-Commissioning 0b1 is sent, there is probably a special action for the user to set the channel on the GPD (e.g. DIP switches).

GOTO step 12 (for proxy-based commissioning) or step 13 (for proximity-based commissioning).

4. **Proxy commissioning state machine:** GPP 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**;

d. Commissioning GPDF or Data GPDF with Auto-Commissioning set to 0b1 – **GOTO step 12**;

e. Commissioning Reply GPDF – **GOTO step 16**;

f. Success GPDF – **GOTO step 17**.

5. **GPS commissioning state machine:** GPS 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 Data GPDF with Auto-Commissioning set to 0b1 – **GOTO step 13**;

e. Commissioning Reply GPDF – **GOTO step 16**;

f. Success GPDF – **GOTO step 18**.

<u>In-band channel determination part</u>

6. **Proxy receives Channel Request GPDF**: GPPs 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.

b. If for whatever reason the GPP cannot act as a TempMaster for this command, (e.g. because it does not support the channel indicated in the Channel Request, or because it is in the middle of an important procedure on the ZigBee network), it may withhold from sending a GP Commissioning Notification

c. If they are in commissioning mode, each GP that can act as a TempMaster forms a GP Commissioning Notification message, with *AppointTempMaster* sub-field of the Options field set to 0b1 and the fields *GPP short address* and *GPP distance* included; 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.

The GP Commissioning Notification is sent as broadcast on the operational channel **with proxy's own address and sequence number**, after *gppTunnelingDelay*, and the scheduled transmission should be dropped only if GPP receives the same frame within *gppTunnelingDelay* forwarded by a different proxy with shorter distance to the GPD, or same distance and lower short address.

7. **GPS receives GPD Channel Request command:** The GPS receives a GPD Channel Request command (either directly or in a GP Commissioning Notification).

a. If NOT in commissioning mode, the GPS silently drops the command.

b. If the GPS received the GPDF in direct mode, and the frame was protected, the GPS shall security-check and security process the incoming packet (as described in sec. A.3.7.2).
If security check and security processing succeed, **GOTO step 7d.**

c. If security processing fails, and also in the case of GPDF received in tunneled mode with *Security processing failed* sub-field of the *Options* field of the GP Commissioning Notification command set to 0b1, the behavior is vendor- and application-specific. **GOTO step 5**.

d. the GPS appoints the TempMaster:

  i. If proxy-based commissioning: the GPS waits for *Dmax* to collect a couple of GP Commissioning Notification commands (from various GPPs), selects the GPP with shortest distance to the GPD and, if many, lowest address

  ii. If the GPS 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.

  iii. If one of the proxies is appointed as a TempMaster, the GPS 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.
  Note: *to improve the robustness of the procedure, the GPS 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.*

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. Other proxies silently drop it 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 (GPP or GPS) receives the Channel Request on channel *TransmitChannel,*

a. If GPP: does NOT send a GP Commissioning Notification, neither on the operational channel nor on *TransmitChannel*;

b. Immediately switches to the Tx mode on channel *TransmitChannel*, and after *gpTxOffset* starts the transmission of the Channel Configuration GPDF; the Channel Configuration GPDF ***may*** be sent unprotected.
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*.

c. TempMaster returns to operational channel in commissioning mode.

If no Channel Request 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 (GPP) or step 5 (GPS)**.

Should the TempMaster receive ANY OTHER GPDF than Channel Request GPDF on channel *TransmitChannel*, including a Commissioning GPDF or Success GPDF, it should silently drop it.

10. **GPD receives Channel Configuration GPDF:** The GPD receives the Channel Configuration GPDF, stores the operational channel, and sets its *ToggleChannel* internal variable to FALSE. **GOTO step 3**.
11. All GPPs and GPS receiving the Channel Configuration GPDF silently drop it. **GOTO step 3**.

<u>Commissioning part</u>

12. **Proxy receives commissioning command:** GPPs (also in GPC) receiving a Commissioning GPDF or Data GPDF with *Auto-Commissioning* = 0b1 on the operational channel:
    a. If they are NOT in commissioning mode: silently drop the Commissioning GPDF.
    a. If the GPDF was protected, all the GPP shall security-check and security-process it (see sec. A.3.7.2, A.1.5.4).
        i. If security processing fails on a GPP, the GPP ***shall*** forward the frame with *Security processing failed* sub-field of the *Options* field of the GP Commissioning Notification set to 0b1.
        ii. Otherwise, if security processing succeeds, the GPP proceeds with step b).
    b. If *RxAfterTx* = TRUE, all GPP check if they have a GPDF in the gpTxQueue for this GPD. If a GPP finds a Commissioning Reply GPDF for this GPD in its gpTxQueue (i.e. it is the TempMaster), its GP stub sends the Commissioning Reply GPDF without CSMA/CA, using the same security level as the triggering GPDF, and starting the transmission *gpTxOffset* after the reception of the triggering GPDF, on the operational channel. The transmission ***shall not*** take longer than *gpTxDuration*.
    *Note: (MAC ACK **shall not** be requested).*
    c. The GPP checks if it already has a Proxy Table entry for this GPD:
        iii. If yes, the settings of the *EntryActive*/*EntryValid* flags remain unchanged; the *InRange* flag is set to 0b1;
        iv. 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.
    d. All GPPs form a GP Commissioning Notification message with *Security processing failed* flag set to 0b0 and all available GPD capability information in the corresponding fields, to be sent on the operational channel,
        i. If GPD *RxAfterTx*=TRUE, the proxy sets the flag *AppointTempMaster* in the GP Commissioning Notification payload Options field to 0b1 and include their short address and distance to the GPD in the then present *GPP short address* and *GPP distance* fields of the GP Commissioning Notification command;
        the GP Commissioning Notification is sent as broadcast **with proxy's own address and sequence number** after *gppTunnelingDelay*, and is to be dropped only if the GPP sees the same frame within *gppTunnelingDelay* forwarded by a different proxy with the *GPP distance* field from the received command has a higher value than the distance measured by the receiving proxy on receipt of this GPDF, or if the distance value is equal, if the value in the *GPP address* field is lower than this proxy's NWK;
        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 GPD *RxAfterTx*=FALSE, they set the flag *AppointTempMaster* in the GP Commissioning Notification payload *Options* field to 0b0, and do not include their short address and distance

for the just received GPDF in the GP Commissioning Notification command;
the GP Commissioning Notification is sent as broadcast using proxy aliasing after *Dmin* (see sec. A.3.6.3.1).
GOTO step 13.

13. **GPS receives commissioning command:** The pairing GPS receives a Commissioning GPDF or Data GPDF with Auto-Commissioning 0b1 on the operational channel (in GP Commissioning Notification command or directly).
    a. If not in commissioning mode, the GPS silently drops the Commissioning GPDF;
    b. If the GPS received the GPDF in direct mode, and the frame was protected, the GPS shall security-check and security process the incoming packet (as described in sec. A.3.7.2, A.1.5.4).
    c. If security processing fails, and also in the case of GPDF received in tunneled mode with *Security processing failed* sub-field of the *Options* field of the GP Commissioning Notification set to 0b1, the behavior is vendor- and application-specific.
    d. If security processing succeeds or if the GPDF was unprotected, GPS checks if the minimum security level supported by the GPD is equal to or larger than *gpsSecurityLevel* (see sec. A.3.3.2.6). If there is no match, the GPS drops the frame; further behavior is vendor- and application-specific.
    e. the GPS checks if GPD application functionality matches (see sec. A.3.6.2.1). If there is no match, the GPS drops the frame; further behavior is vendor- and application-specific.
    f. If GPD application functionality matches, the GPS shall check the contents of the security-related fields of the Commissioning GPDF (see sec. A.1.5.4)
        i. If the check fails the behavior is vendor- and application-specific.
        ii. If the check succeeds, the GPS stores the supplied GPD capability information, including the security-related parameters in a Sink Table entry and continues with step (g).
        Note: If the commissioning command is a Data GPDF with *Auto-Commissioning* flag set to 0b1, the GPS shall use the following default values: *MAC sequence number capability* = 0b1; *RxOnCapability* = 0b0; *FixedLocation* = 0b0; if the GPDF was protected, the *SecurityLevel* and *SecurityKey* used, otherwise *SecurityLevel* = 0b00 and *KeyType* = 0b000.
    g. If the GPS already had a Sink Table entry for this GPD, the GPS 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.
    h. If Data GPDF with *Auto-Commissioning* 0b1 or Commissioning GPDF with *RxAfterTx*=FALSE – **GOTO step 19**.
    i. Else if Commissioning GPDF with *RxAfterTx*=TRUE,
        i. GPS prepares the Commissioning Reply GPDF, carrying the parameters requested by the GPD in the Commissioning GPDF. If none are requested, but *RxAfterTx*=TRUE, Commissioning Reply GPDF ***shall*** still be created, with only the *Options* field present.
        ii. GPS appoints the TempMaster:
            - If proxy-based commissioning: the GPS waits for *Dmax* to collect a couple of GP Commissioning Notification commands (from various GPP), selects the selects TempMaster as described in sec. A.3.6.2.3;
            - If the GPS 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 GPS 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**.

14. **GP Response carrying GPD Commissioning Reply command**: GPPs receiving the GP Response command with the Commissioning Reply (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 Commissioning Reply GPDF (taking the supplied GPD Commissioning Reply 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.
    d. **GOTO step 3**.

15. **GPD receives Commissioning Reply GPDF:** A GPD receiving a Commissioning Reply GPDF:
    a. checks if 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).
    c. Sets the *ParametersStored* flag to TRUE **GOTO step 3**.
16. All GPPs and GPSs receiving a Commissioning Reply GPDF ignore it. **GOTO step 3**.

17. **Proxy receives Success GPDF:** GPPs (also in GPC) receiving a Success GPDF:
    a. If they are NOT in commissioning mode: silently drop the Success GPDF.
    b. If the Success GPDF was protected, all the GPP shall security-check and security-process it (see sec. A.3.7.2, A.1.5.4).
        i. If security processing fails on a GPP, the GPP ***shall*** forward the frame with *Security processing failed* flag set to 0b1.
        ii. Otherwise, if security processing succeeds, the GPP proceeds with step (c).
    c. All GPPs form a GP Commissioning Notification message, to be sent on the operational channel, containing the GPD Success command ID (0xE3) in the *GPD Command ID* field and 0xff in the *GPD Command payload* field.
       Since GPD *RxAfterTx*=FALSE, they clear the flag *AppointTempMaster* in the GP Commissioning Notification payload *Options* field, and do not include their short address and distance for the just received GPDF in the GP Commissioning Notification command;
       the GP Commissioning Notification is sent as broadcast using proxy aliasing after Dmin (see sec. A.3.6.3.1).
       GOTO step 18.

18. **GPS receives Success GPDF:** GPS receiving a GPD Success command:
    a. If the GPS is NOT in commissioning mode: silently drop the Success GPDF.
    b. The Success GPDF ***shall*** be protected as agreed for the operational mode of this GPD.
       The GPS ***shall*** security-check and security-process it (see sec. A.3.7.2, A.1.5.4).

   i. If security processing fails, the commissioning failed. The behavior is vendor- and application-specific.
   ii. Otherwise, if security processing succeeds, the GPS proceeds with **step 19**.

<u>**Commissioning finalization**</u>

19. **GPS finalizes commissioning:** Pairing GPS:
   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.
   c. If GPS supports Translation Table functionality: not done before and if the sink does not have default generic GPD Command Translation Table entries for all GPD Data commands supported by this GPD, the GPS creates default Translation Table entries for all GPD Data commands supported by this GPD (see Table 52).
   d. If required, assigns an AssignedAlias for the GPD.
   e. Sends Device_annce for the alias (derived or assigned) for the GPD.
   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 GPS adds its GPEP to the corresponding APS group.
      If the security level is > 0b00, the GPS ***shall*** include the *GPD key* field in the GP Pairing command, irrespective of the key type.
   g. If the GPS is a GPT/GPT+/GPC, the GPS ***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 GPS 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.
      GPT/GPT+/GPC ***shall not*** send GP Pairing Configuration command for unicast or derived groupcast pairing.
   h. If the GPS is a GPCm (and/or supports *SinkTable-based groupcast forwarding* functionality), the GPS ***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 GPCm 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.
   i. (if required) the user puts the GPS into operational mode
   j. (if required) GPS 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 GPSs 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 GPPs receiving the GP Pairing
    a. create/update Proxy Table entry
    b. optionally, exit commissioning mode (if that was the *ExitMode* condition). **GOTO step 22**.

22. GPPs receiving GP Proxy Commissioning Mode with *Action* sub-field of the *Options* field set to 0b0 = exit (if sent) switch back to operational mode. **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.

In Figure 69 and Figure 70 an exemplary message sequence chart for proxy-based commissioning of a GPD with RxOnCapability is depicted.

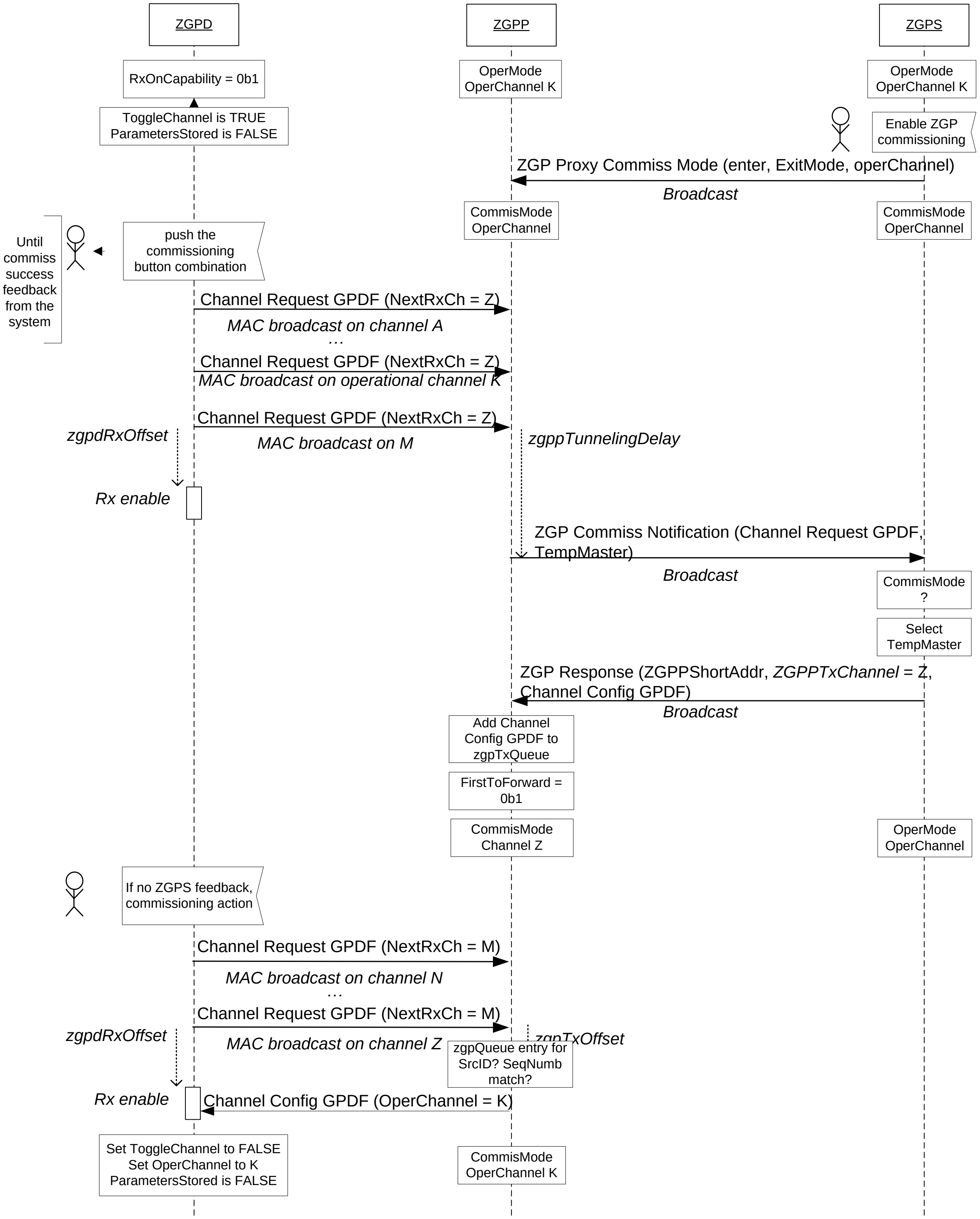


Figure 69 – Exemplary MSC for proxy-based commissioning for bidirectional commissioning capable GPD (part 1)

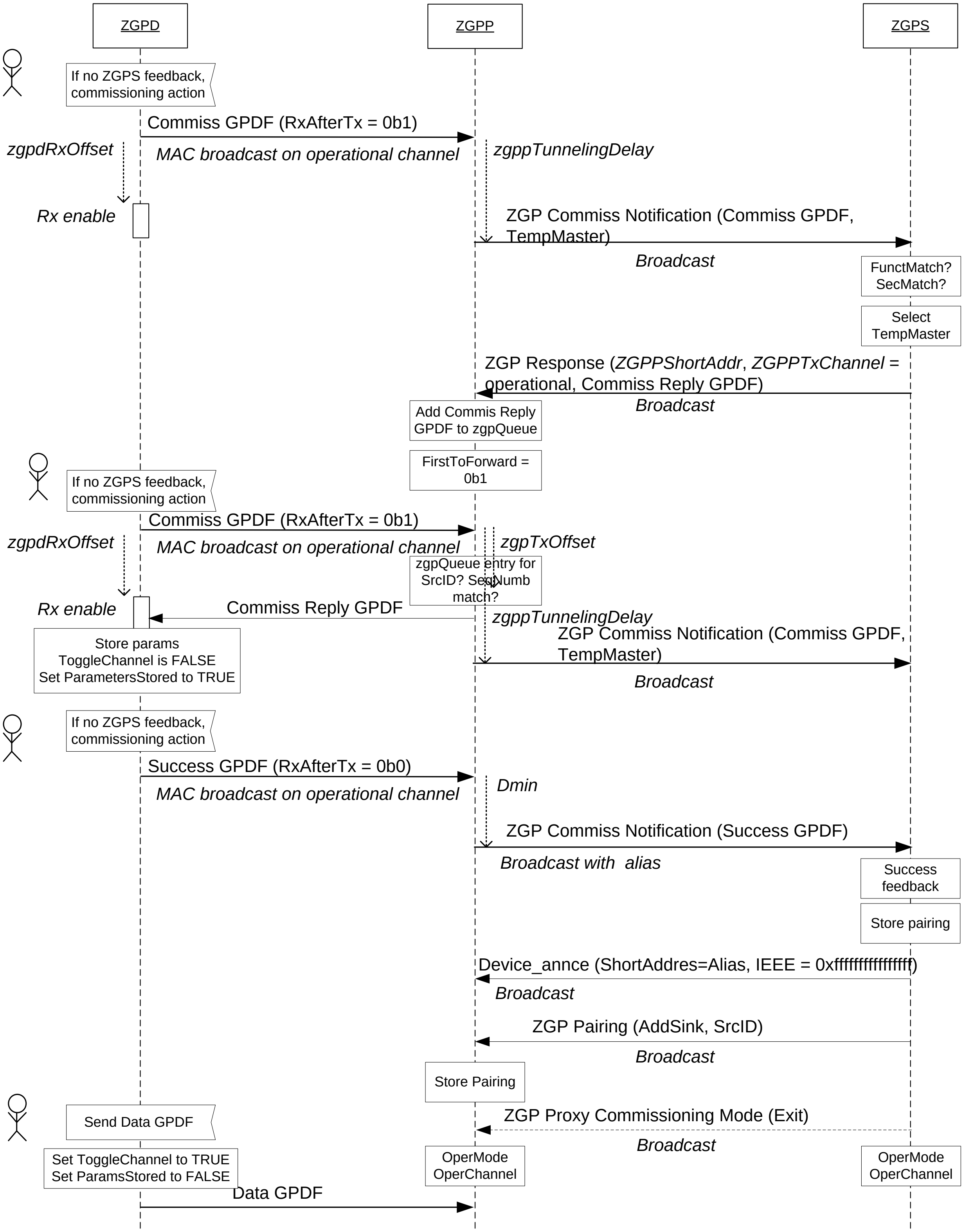

ZGPD
ZGPP
ZGPS
If no ZGPS feedback, commissioning action
Commiss GPDF (RxAfterTx = 0b1)
zgpdRxOffset
MAC broadcast on operational channel
zgppTunnelingDelay
Rx enable
ZGP Commiss Notification (Commiss GPDF, TempMaster)
Broadcast
FunctMatch? SecMatch?
Select TempMaster
ZGP Response (ZGPPShortAddr, ZGPPTxChannel = operational, Commiss Reply GPDF)
Broadcast
Add Commis Reply GPDF to zgpQueue
FirstToForward = 0b1
If no ZGPS feedback, commissioning action
Commiss GPDF (RxAfterTx = 0b1)
zgpdRxOffset
MAC broadcast on operational channel
zgpTxOffset
zgpQueue entry for SrcID? SeqNumb match?
Rx enable
Commiss Reply GPDF
zgppTunnelingDelay
ZGP Commiss Notification (Commiss GPDF, TempMaster)
Store params ToggleChannel is FALSE Set ParametersStored to TRUE
Broadcast
If no ZGPS feedback, commissioning action
Success GPDF (RxAfterTx = 0b0)
Dmin
MAC broadcast on operational channel
ZGP Commiss Notification (Success GPDF)
Broadcast with alias
Success feedback
Store pairing
Device_annce (ShortAddres=Alias, IEEE = 0xffffffffffffffff)
Broadcast
ZGP Pairing (AddSink, SrcID)
Broadcast
Store Pairing
ZGP Proxy Commissioning Mode (Exit)
Send Data GPDF
Broadcast
Set ToggleChannel to TRUE Set ParamsStored to FALSE
OperMode OperChannel
OperMode OperChannel
Data GPDF

Figure 70 – Exemplary MSC for proxy-based commissioning for bidirectional commissioning capable GPD (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 GPP

When GPP receives in commissioning mode:

- an unprotected Data GPDF with *Auto-Commissioning* sub-field set to 0b1 or unprotected Commissioning GPDF; the GPP 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, *Security processing failed* set to 0b0.
- a protected Data GPDF with *Auto-Commissioning* sub-field set to 0b1 or protected Commissioning GPDF:
  - and the GPP has the key and security processing succeeds (see A.3.7.2.1), the GPP 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, *Security processing failed* 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 GPP has the key, but the security processing fails (see A.3.7.2.1), the GPP 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; *Security processing failed* 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 (for SecurityLevel 0b01, pre-padded with zeros); *GPD security Frame Counter* carrying the value copied from the GPDF (for SecurityLevel 0b01, pre-padded with zeros).
    The Proxy Table entry ***shall not*** be updated with the new *GPD security Frame Counter* value.
- the GPP does not have the key, it ***should*** drop the GPDF.

A.3.9.2.1.2 GPS

The following applies to GPD command used for commissioning, either received directly or tunneled in the GP Commissioning Notification with *Security processing failed* 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 (*GP security key request*, *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 GPS. 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 *GP security key request* (and *RxAfterTx*) was also set, the GPS ***shall not*** include the key in GPDF Commissioning Reply frame.
  - 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 *RequestGPSecurityKey* (and *RxAfterTx*) was also set, the GPS ***may*** include the key in GPDF Commissioning Reply frame.
  - If the *KeyType* sub-field of the *Extended Options* field has the value of 0b000, and the *GP security key request* (and *RxAfterTx*) is also set, the check succeeds. The GPS ***shall*** include the key in GPDF Commissioning Reply frame.
  - If the *GP security key request* was set to 0b1, but *RxAfterTx* was set to 0b0, or if *GP security key request* was set to 0b1, but *SecurityLevelCapabilities* was set to 0b0, the check fails.

The behaviour on check failure as in the cases listed above and on reception of GP Commissioning Notification with *Security check failed* sub-field set to 0b1, is application-specific and out-of-scope of this document.

A.3.9.2.2 GPD commissioning

The GPD that supports security (*SecurityLevelCapabilities* > 0b00) has the following security configuration options for commissioning mode:

- If the GPD is capable of sending the Success GPDF and if in the commissioning process the GPD and the pairing GPS 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 agreed is *gpSecurityLevel*=0b11, the GPD ***shall*** protect the Success GPDF using either *gpSecurityLevel*=0b10, if supported, or *gpSecurityLevel*=0b11;
- If the GPD is capable of sending the Commissioning GPDF and:
  - the GPD has the NWK key (*gpSecurityKeyType* = 0b001) or a GPD group key *gpSecurityKeyType* = 0b010 or 0b011), the Commissioning GPDF ***should*** be sent protected, using the agreed *gpSecurityLevel*; the sub-fields *SecurityLevel* and *SecurityKey* of the *Extended NWK Frame Control* field ***shall*** be set accordingly. In the Commissioning command payload, the *GPDkey* field ***shall not*** be present, but 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 0b0 and 0b1, respectively.
  - 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; the corresponding sub-fields ***shall*** be set accordingly; the TC-LK protection ***may*** be used.
  - the GPD has no key, the Commissioning GPDF ***shall*** be sent unprotected, and in the

    Commissioning command payload, the *GPDkey* field ***shall not*** be present and the *Security Frame Counter* field ***shall*** be present.

  - (in addition to any of the options above) the GPD has the energy for receiving Commissioning Reply GPDF containing a key, and wishes to request it, it ***shall*** also set the *GPD security key request* sub-field of the *Options* field of the Commissioning GPDF to 0b1; and the *RxAfterTx* sub-field of the *Extended NWK Frame Control* field to 0b1.
    Note: Overwriting the individual key by the GPS requires the GPD to first send and then receive a long GPDF with the 16B security key.

- 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. A.1.5.3.3.

A.4GreenPower cluster extensions: ApplicationID 0b000 and 0b010

A.4.1 GPD CommandIDs

Table 48 and Table 49 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.

Note: Groups commands and Add Scene / Remove Scene / Remove All Scenes are managed through a specific configuration frame. Only View Scene commands are made directly available to GP devices. The allocation below assumes that GP devices can support up to 16 scenes.

The command range 0xf0 – 0xff is reserved for commands sent to the GPD. They are defined in Table 50.

Table 48 – 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	Scene 0	Scenes	View Scene	0
0x11	Scene 1	Scenes	View Scene	1
0x12	Scene 2	Scenes	View Scene	2
0x13	Scene 3	Scenes	View Scene	3
0x14	Scene 4	Scenes	View Scene	4
0x15	Scene 5	Scenes	View Scene	5
0x16	Scene 6	Scenes	View Scene	6
0x17	Scene 7	Scenes	View Scene	7
0x18	Scene 8	Scenes	View Scene	8
0x19	Scene 9	Scenes	View Scene	9
0x1A	Scene 10	Scenes	View Scene	10
0x1B	Scene 11	Scenes	View Scene	11
0x1C	Scene 12	Scenes	View Scene	12
0x1D	Scene 13	Scenes	View Scene	13
0x1E	Scene 14	Scenes	View Scene	14
0x1F	Scene 15	Scenes	View Scene	15
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			

GPD command		Mapping to ZigBee		
CommandID	Command Name	Corresponding ClusterID	CommandID	Command Payload
0x30 – 0x33	Defined in Table 49			
0x34	Level Control/Stop	Level Control	Stop	N/A
0x35 – 0x38	Defined in Table 49			
0x39 – 0x3F	Reserved			
0x40	Move Hue Stop	Color Control	Move Hue	Stop
0x41 – 0x44	Defined in Table 49			
0x45	Move Saturation Stop	Color Control	Move Saturation	Stop
0x46 – 0x4B	Defined in Table 49			
0x4C – 0x4F	Reserved			
0x50	Lock Door	Door Lock	Lock Door	N/A
0x51	Unlock Door	Door Lock	Unlock Door	N/A
0x52 – 0x5F	Reserved			
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		
0x69-0x6f	Reserved			
0x70-0x9f	Reserved			
0xA0-0xE0	Defined in Table 49			
0xE1	Decommissioning	N/A		
0xE2	Success	N/A		
0xE3	Defined in Table 49			
0xE4-0xEF	Defined in Table 49			

Table 49 defines CommandIDs for commands with non-zero payload, for *ApplicationID* of 0b000 and 0b010.

Table 49 – 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	

GPD command		Mapping to ZigBee		
CommandID	Command Name	ClusterID	Command Name	Command payload
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	
0x4A	Move Color	Color Control	Move Color	
0x4B	Step Color	Color Control/	Step Color	
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 Request 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 – 0xAE	Reserved			
0xAF	Any GPD sensor command (0xA0 – 0xA3)	Copied from the triggering GPD command	ZCL Report attributes command	Copied from the triggering GPD command
0xB0-0xDF	Reserved			
0xE0	Commissioning	N/A		
0xE3	Channel Request	N/A		
0xE4 – 0xEF	Reserved			

Table 50 – 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 – 0xFF	Reserved for other commands sent to the GPD			

A.4.2 Format of individual commands

A.4.2.1 Commissioning commands

A.4.2.1.1 Commissioning

The payload of the Commissioning GPD command is formatted as shown in Figure 71.

1	1	0/1	0/16	0/4	0/4
8-bit enumeration	8-bit bitmap	8-bit bitmap	Security Key	Unsigned 32-bit integer	Unsigned 32-bit integer
GPD DeviceID	Options	Extended Options	GPD Key	GPD Key MIC	GPD outgoing counter

Figure 71 – Format of the Commissioning command payload

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.[3]

The *Auto-Commissioning* sub-field of the *NWK Frame Control* field for the Commissioning GPDF shall always be set to 0b0. The *GPD CommandID* field shall carry the value 0xE0, indicating the Commissioning command, as defined in Table 49.

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 Table 51.

A.4.2.1.1.2 Options field

The *Options* field of the Commissioning GPDF has the format as specified in Figure 72.

Bits: 0	1	2	3	4	5	6	7
MAC sequence number capability	RxOnCapability	Reserved	Reserved	PANId request	GP Security Key request	Fixed Location	Extended Options field

Figure 72 – Format of the Options field of the 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 *Reserved* sub-fields, if set, ***shall*** be ignored by the devices according to the current version of the specification. The other fields and sub-fields as defined in the current version of the specification ***shall*** be processed. [4]

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

[3] CCB #1661, as resolved in GP v1.0 errata, 12-0624r00

[4] CCB #1661, as resolved in GP v1.0 errata, 12-0624r00

ignored on reception, if the *RxAfterTx* sub field of the *NWK Frame Control* field of the Commissioning GPDF is set to 0b0.

The *GP Security Key request* 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 Commissioning GPDF is set to 0b0.

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 *Extended Options Field* 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 Commissioning GPDF has the format as specified in Figure 73.

Bits: 0-1	2-4	5	6	7
SecurityLevel capabilities	KeyType	GPD Key present	GPD Key encryption	GPD outgoing counter present

Figure 73 – Format of the *Extended Options* field of the Commissioning command

The *SecurityLevelCapabilities* sub-field indicates the device’s security capabilities during normal operation. It can take values as defined in A.1.5.3.2.

When the *Extended Options* field is not present in the Commissioning GPDF and the *GP Security Key request* 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 Commissioning GPDF and the *GP Security Key request* 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 Table 14.

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 in A.1.5.3.3.3; and the *GPDkeyMIC* field contains the MIC for the encrypted GPD key, calculated as described in A.1.5.3.3.3. 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.1.5.3.3.3; 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 *GPD outgoing counter present* 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.

A.4.2.1.1.4 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.4.2.1.1.5 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 74.

Octets	**1**	**0/2**	**0/16**	**0/4**
Data Type	8-bit bitmap	Unsigned 16-bit integer	Security key	Unsigned 32-bit integer
Field name	Options	PANId	GPD Security Key	GPD Key MIC

Figure 74 – Format of the 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 50.

A.4.2.1.2.1 Options field

The *Options* field is formatted as shown in Figure 75.

Bits: 0	**1**	**2**	**3-4**	**5-7**
PANID present	GPD security key present	GPD key encryption	SecurityLevel	KeyType

Figure 75 – Format of the Options field of 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 *GPD Security Key present* 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 *GPD Security Key* 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.1.5.3.3.3; and the *GPDkeyMIC* field contains the MIC for the encrypted GPD key, calculated as described in A.1.5.3.3.3. When the *GPD Security Key present* sub-field is set to 0b0, the *GPDKeyEncryption* sub-field is ignored.

If the *SecurityLevel* sub-field is set to 0b00, the *GPD Security Key* field is not present and the sub-fields *GPD key encryption* 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 carried in the *SecurityKey* field, and can take values as defined in A.1.5.3.3.

A.4.2.1.2.2 When generated

The GPD Commissioning Reply command is generated by the commissioning GPS 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 ***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, GPPs act as described in A.3.5.2.1, and the GPSs 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 76.

Octets	**1**
Data Type	8-bit bitmap
Field name	*Channel toggling behavior*

Figure 76 – Format of the Channel Configuration command payload

The *Channel Toggling Behavior* field is formatted as shown in Figure 77.

Bits: 0-3	**4-7**
Rx channel in the next attempt	Rx channel in the second next attempt

Figure 77 – Format of the Channel Toggling Behavior field of the 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. If Channel Request GPDF is to be sent secured, it ***shall*** be sent with *Frame Type* 0b00; then, for *ApplicationID* = 0b000the *GPD SrcID* field, and for *ApplicationID* = 0b010 the IEEE address field ***shall*** be present. To minimize the length of Channel Request, *Frame Type* 0b01 ***may*** be used.

A.4.2.1.5 Channel Configuration command

The payload of the Channel Configuration command is formatted as shown in Figure 78.

Octets	1
Data Type	8-bit bitmap
Field name	*Channel*

Figure 78 – Format of the Channel Configuration command payload

The *Channel* field is formatted as shown in Figure 79.

Bits: 0-3	4-7
Operational Channel	Reserved

Figure 79 – Format of the Channel field of the Channel Configuration command

The *OperationalChannel* sub-field can take the following values: 0b0000: channel 11, 0b0001: channel 12, …, 0b1111: channel 26.

The Channel Configuration GPDF can use the following values of the *Frame Type* sub-field of the *NWK Frame Control* field: 0b01 and 0b00. If Channel Configuration GPDF is to be sent secured, it ***shall*** be sent with *Frame Type* 0b00; then, for *ApplicationID* = 0b000 the *GPD SrcID* field, and for ApplicationID = 0b010 the IEEE address field ***shall*** be present. Then, the *Direction* sub-field of the *Extended NWK Frame Control* field shall be set to 0b1.

To minimize the length of Channel Request, *Frame Type* 0b01 ***may*** be used.

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.4.2.3 Sensor commands

All sensor commands defined here ***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 Commissioning GPDF (see sec. A.4.2.1.1).

A.4.2.3.1 Attribute reporting command

The command payload for the Attribute reporting command is formatted as shown in Figure 80.

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 80 – Payload of the 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 81.

Octets	**2**	**1**	**variable**
Field name	AttributeID	Attribute data type	Attribute data

Figure 81 – 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 Manufacturer-specific attribute reporting command is formatted as shown in Figure 82.

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 82 – Payload of the 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 [8].

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 81.

A.4.2.3.3 Multi-cluster reporting command

The command payload for the Multi-cluster reporting command is formatted as shown in Figure 83.

Octets	variable	variable	…	variable
Data Type	structure	structure	…	structure
Field name	Cluster report 1	Cluster report 2	…	Cluster report n

Figure 83 – Payload of the Multi-cluster reporting command

Cluster report field shall be formatted as depicted in Figure 84.

Octets	2	2	1	variable
Field name	ClusterID	AttributeID	Attribute data type	Attribute data

Figure 84 – 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.

A.4.2.3.4 Manufacturer-specific multi-cluster reporting command

The command payload for the Manufacturer-specific multi-cluster reporting command is formatted as shown in Figure 85.

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 85 – Payload of Manufacturer-specific multi-cluster reporting command

The *Manufacturer Code* carries the Manufacturer ID. It can take values as defined in [8].

Cluster report field shall be formatted as depicted in Figure 84. 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.

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 86.

Octets	**0/1**
Data Type	Unsigned 8-bit integer
Field name	Rate

Figure 86 – Payload the 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. If the *Rate* field is 0xff the device should move as fast as it is able. If the device is not able to move at a variable rate, this field may be disregarded.

The presence of the *Rate* field is optional, and can be deduced from the command payload length. If the *Rate* field is not present, then the receiver shall move as fast as it is able, if it has a variable rate, or else at the only available rate.

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 86.

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 87.

Octets	**1**	**0/2**
Data Type	Unsigned 8-bit integer	Unsigned 16-bit integer
Field name	Step size	Transition time

Figure 87 – Payload the 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. If the *Transition time field* is 0xffff the device should move as fast as it is able. 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. If the *Transition time* field is not present, then the receiver shall move as fast as it is able, 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 87.

A.4.2.4.5 'With On/Off' Commands

The Move Up/Down (with On/Off) and Step Up/Down (with On/Off) commands have identical payloads to the Move Up/Down and Step Up/Down commands, 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.
- 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 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 86.

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, then the receiver shall move as fast as it is able, 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 87.

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, then the receiver shall move as fast as it is able, 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 86.

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, then the receiver shall move as fast as it is able, 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 87.

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, then the receiver shall move as fast as it is able, 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 88.

Octets	**2**	**2**
Data Type	Signed 16-bit integer	Signed 16-bit integer
Field name	RateX	RateY

Figure 88 – Payload of the 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 89

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 89 – Payload the 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, then the receiver shall move as fast as it is able, 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 90.

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 90 – Payload of the Request Attributes command

The *Options* field is formatted as shown in Figure 91.

Bits: 0	1	2..7
Multi-record	Manufacturer field present	Reserved

Figure 91 – Format of the Options field of the 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 [8].

The *Cluster Record Request* field is formatted as shown in Figure 92. 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 92 – 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 command payload for the Read Attributes Response command is formatted as shown in Figure 93.

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 93 – Payload of the Read Attributes Response command

The *Options* field is formatted as shown in Figure 91, and the sub-fields are defined as in A.4.2.6.1.

The *Manufacturer ID* field can take values as defined in [8].

The *Cluster record* field is formatted as shown in Figure 94.

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 94 – 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 95.

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 95 – 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 command payload for the Write Attributes command is formatted as shown in Figure 96.

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 96 – Payload of the Write Attributes command

The Options field is formatted as shown in Figure 91, and the subfields are defined as in A.4.2.6.1.

The *Manufacturer ID* field can take values as defined in [8].

The *Write cluster record* field is formatted as shown in Figure 97.

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 97 – 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 98.

Octet: 2	1	Variable
Unsigned 16-bit integer	8-bit enumeration	variable
AttributeID	Attribute Data Type	Attribute Value

Figure 98 – 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 90, Figure 91, and Figure 92.

A.4.3 GP Devices (GPD)

Table 51 lists GP Devices for the ApplicationID sub-field of the Extended NWK Frame Control field set to 0b000 or 0b010. For the definition of GP Devices, please see [4].

GP Devices (GPD), i.e. the energy-harvesting devices, have their own device descriptions and IDs, 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).

Different definitions and IDs are chosen for GP devices, because they have a different set of mandatory and optional clusters as their normal ZigBee counterparts. This also allows for additional flexibility in defining devices in the future that will only work with energy harvesters. For efficiency, this limited set of identifiers is encoded on 1 byte.

Device descriptions specified in the GP profile extension are summarized in Table 51, - along with their proposed respective Device IDs.

Additional devices may be added in the future.

Table 51 – List of GPDs for ApplicationID 0b000 and 0b010

	Device	GPD Device ID
GP Generic	GP Simple Generic 1-state Switch	0x00
	GP Simple Generic 2-state Switch	0x01
	GP On/Off Switch	0x02
	GP Level Control Switch	0x03
	GP Simple Sensor	0x04
	GP Advanced Generic 1-state Switch	0x05
	GP Advanced Generic 2-state Switch	0x06
	Reserved	0x07 – 0x0F
GP Lighting	GP Color Dimmer Switch	0x10
	GP Light Sensor	0x11
	GP Occupancy Sensor	0x12
	Reserved	0x13 – 0x1f
GP Closures	GP Door Lock Controller	0x20
	Reserved	0x21 – 0x2F
GP HVAC	GP Temperature Sensor	0x30
	GP Pressure Sensor	0x31
	GP Flow Sensor	0x32
	GP Indoor Environment Sensor	0x33
	Reserved	0x34 – 0x3F
	Reserved	0x40 – 0xFF

The minimal GPD application functionality (GPD CommandIDs) supported by particular types of GPDs according to this specification is defined in Table 52 and Table 53 below.

Table 52 – List of GPD commands per GPD

	Device	Transmitted GPD commands (Command IDs as in Table 48 and Error! Reference source not found.)	
		Mandatory	Optional
GP Generic	GP Simple Generic 1-state Switch	0x60, 0x61	
	GP Simple Generic 2-state Switch	0x62 - 0x65	
	GP On/Off Switch	0x20-0x21 OR 0x22	0x23
	GP Level Control Switch	0x30, 0x3, 0x34 OR 0x32, 0x33 OR 0x35, 0x36, 0x34 OR 0x37, 0x38	
	GP Simple Sensor	0xA0 or 0xA1 0xE0	
	GP Advanced Generic 1-state Switch	0x60, 0x61, 0x66	
	GP Advanced Generic 2-state Switch	0x62 - 0x65, 0x67, 0x68	

GP Lighting	GP Color Dimmer Switch	At least 1 of 0x40-0x4B;
		0x41 and 0x42: both or none;
		0x43 and 0x44: both or none;
		0x46 and 0x47: both or none;
		0x48 and 0x49: both or none;	

	GP Light Sensor	At least 1 of 0xA0-0xA3 0xE0	
	GP Occupancy Sensor	At least 1 of 0xA0-0xA3 0xE0	
GP Closures	GP Door Lock Controller	0x50-0x51	
GP HVAC	GP Temperature Sensor	At least 1 of 0xA0-0xA3 0xE0	
	GP Pressure Sensor	At least 1 of 0xA0-0xA3 0xE0	
	GP Flow Sensor	At least 1 of 0xA0-0xA3 0xE0	
	GP Indoor Environment Sensor	At least 1 of 0xA0-0xA3 0xE0	

Table 53 defines the ZigBee attributes to be mandatorily supported by the GPD devices using the GPD sensor reporting commands A0-A3. In the second column, it lists the attributes mandatory to report per device type. In the second column, it lists the attributes that must be readable if the GPD has bidirectional operation capability; unless explicitly stated otherwise, they are all read only. The format of all those attributes is defined in the ZCL [3].

1 **Table 53 – List of ZigBee attributes per GPD**

Device	Report Mandatory Attribute	In case of Rx Capability: support mandatory attributes for attribute request (Read only)
GP Simple Sensor	0x0055: PresentValue from Binary Input Cluster	0x0051 Out of Service (Read and Write), 0x0055: PresentValue 0x006F: Status Flags (all from Binary Input Cluster)
GP Light Sensor	0x0000: MeasuredValue from Illuminance Measurement Cluster	0x0000: MeasuredValue 0x0001: MinMeasuredValue, 0x0002: MaxMeasuredValue (all from Illuminance Measurement Cluster)
ZPG Occupancy Sensor	0x0000: Occupancy from Occupancy Sensing Cluster	0x0000: Occupancy 0x0001: OccupancySensorType (all from Occupancy Sensing Cluster)
GP Temperature Sensor	0x0000: MeasuredValue from Temperature Measurement Cluster	0x0000: MeasuredValue 0x0001: MinMeasuredValue 0x0002: MaxMeasuredValue (all from Temperature Measurement Cluster)
GP Pressure Sensor	0x0000: MeasuredValue from Pressure Measurement Cluster	0x0000: MeasuredValue from Pressure Measurement Cluster
GP Flow Sensor	0x0000: MeasuredValue from Flow Measurement Cluster	0x0000: MeasuredValue 0x0001: MinMeasuredValue 0x0002: MaxMeasuredValue (all from Flow Measurement Cluster)
GP Indoor Environment Sensor	0x0000: MeasuredValue from Temperature Measurement Cluster, 0x0000: MeasuredValue from Relative Humidity Measurement Cluster, 0x0000: MeasuredValue from Illuminance Measurement Cluster. For CO2 cluster, as long as the cluster is not part of the ZCL, use manufacturer-specific reporting	0x0000: MeasuredValue 0x0001: MinMeasuredValue 0x0002: Max MeasuredValue (from all of the following clusters: Temperature Measurement Cluster, Relative Humidity Cluster, and Illuminance Cluster).

2