UG392: Using Silicon Labs Green Power with Zigbee EmberZNet PRO

This application note introduces the Silicon Labs Green Power components within the Zigbee EmberZNet PRO stack and explains how to enable a network for Green Power. This document assumes readers are familiar with the basic Green Power concepts discussed in *UG103.15: Silicon Labs Green Power Fundamentals*.

KEY POINTS

- Introduces Green Power.
- Describes a basic Green Power network.
- Explains Green Power commissioning and operational model.
- Describes the Silicon Labs Green Power devices, their plugins, and callbacks.

1. Introduction to Green Power

Zigbee® Green Power (ZGP) is included in the Zigbee 3.0 specification (Z3) (Zigbee Alliance, Zigbee 3.0 specification). It is an end-to-end open standard that allows ultra-low power devices called Green Power Devices (GPDs) to operate on Zigbee networks.

Green Power is actually a number of technologies combined into one global standard that includes these features (Zigbee Green Power, Cam Williams):

- Energy harvesting technology.
  - Includes mechanical, heat, light, pressure (piezo), and other energy harvesting technology.
- Ultra-low power RF silicon that uses many orders of magnitude less power than required for a sleepy or fully networked wireless connection.
  - Uses ultra-low power, non-volatile memory such as Ferroelectric RAM (FRAM).
- An open, global standard network technology that saves even more energy by reducing packet length, round trips, connection rediscovery, and on-network time for devices that may be offline for extended periods of time.
  - Zigbee – 15.4 – 2.4Ghz modules
- An open, global, standard application layer protocol that supports compressed messages and limited transactions using:
  - Zigbee Application Framework (AF)
  - Zigbee Cluster Library (ZCL)

2. Basic Green Power Network

A basic Green Power (GP) network consists of three devices:

- Green Power Device (GPD)
- A Z3 Proxy or Green Power Proxy (GPP)
- A Green Power Sink (GPS)

GPD Frames (GPDF) are transmitted by the GPD devices and received by a Proxy or a Combination (Sink and Proxy) device. The GPP will then encapsulate the received GPDF within a standard Zigbee frame and forward the GPDF packets across the Zigbee PRO / Z3 network in the form of notifications to the Sink that that has been paired with the end device. In a Combination device, the Proxy side is responsible for forwarding the GPDF packets. The following figure illustrates the data flow from the GPD to the GPP and finally to the GPS.

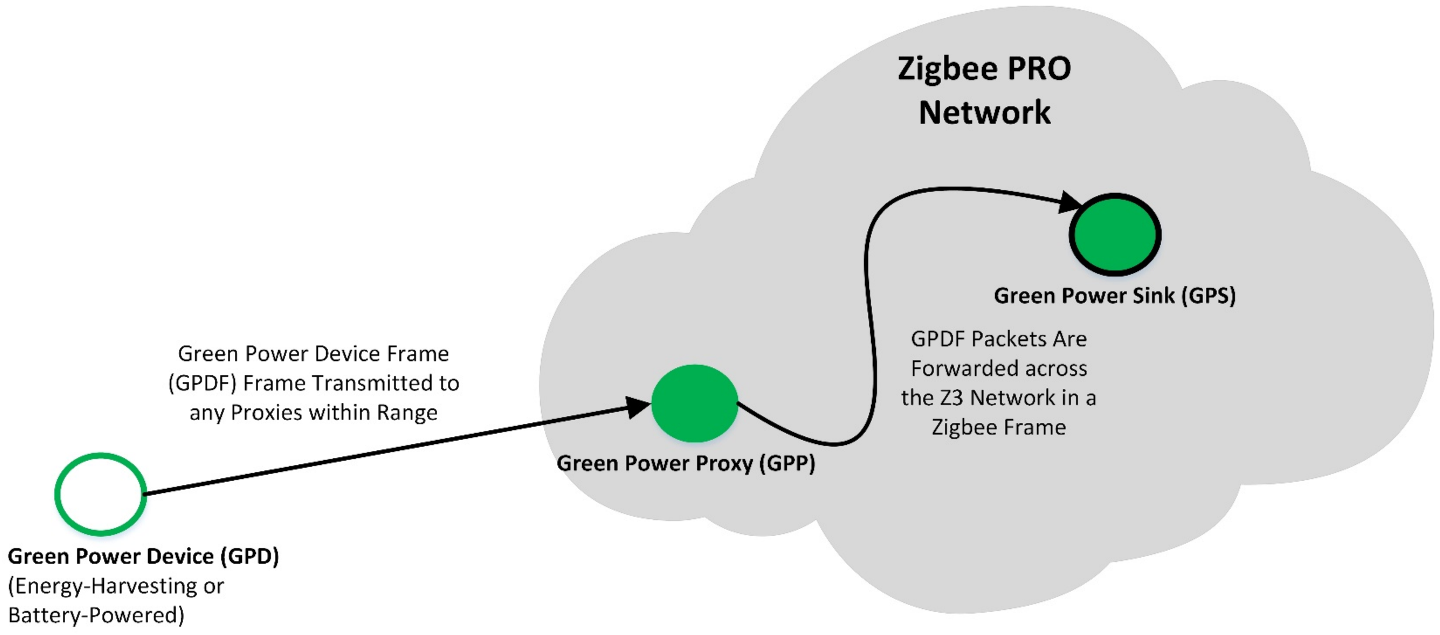

Zigbee PRO Network
Green Power Sink (GPS)
GPDF Packets Are Forwarded across the Z3 Network in a Zigbee Frame
Green Power Device Frame (GPDF) Frame Transmitted to any Proxies within Range
Green Power Proxy (GPP)
Green Power Device (GPD)
(Energy-Harvesting or Battery-Powered)

Figure 2.1. Basic Green Power Message Transmission

As indicated in the following figure, the GPDF is shorter than a standard Zigbee frame (indicated by the dashed line). This allows a GPD to transmit a GPDF using less power than a standard Zigbee frame as the radio transmitter is active for less time.

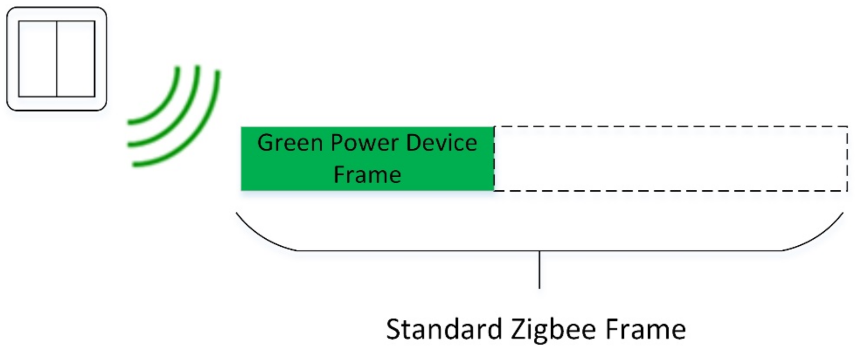


Figure 2.2. GPDF Size

GPDs are primarily one-way devices once in use, although they may optionally (if designed) support bidirectional data exchange during pairing. GPDs should not be considered end devices and Zigbee considers them as less than Zigbee End Devices (ZEDs). For more information on ZEDs, see *UG103.2: Zigbee Fundamentals*.

2.1 How Does Green Power Fit in a Zigbee Network?

The Zigbee stack architecture is illustrated in the following figure.

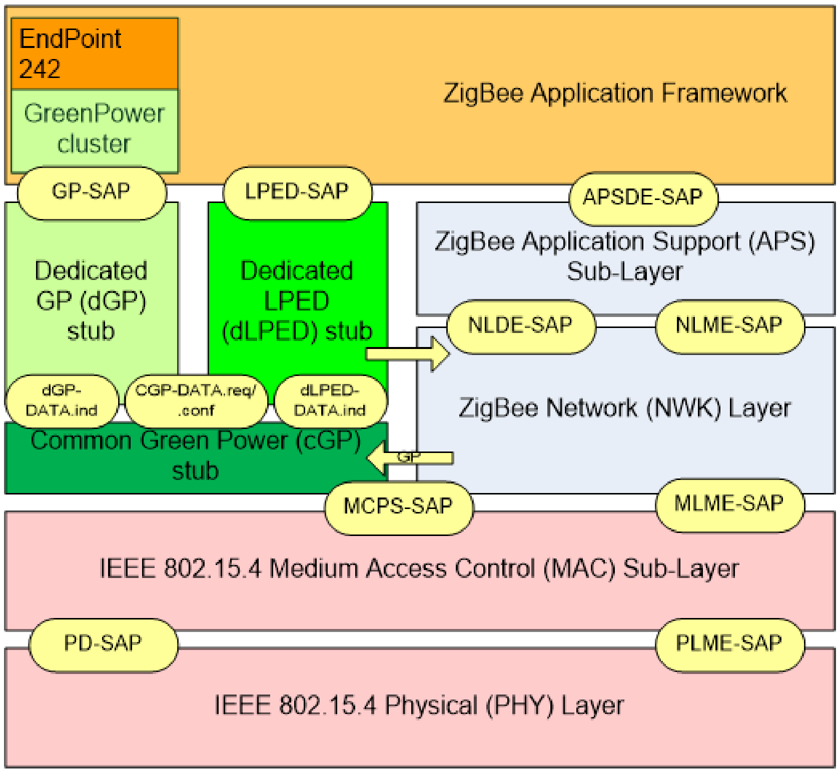

EndPoint 242
GreenPower cluster
ZigBee Application Framework
GP-SAP
LPED-SAP
APSDE-SAP
Dedicated GP (dGP) stub
Dedicated LPED (dLPED) stub
ZigBee Application Support (APS) Sub-Layer
NLDE-SAP
NLME-SAP
dGP-DATA.ind
CGP-DATA.req/.conf
dLPED-DATA.ind
ZigBee Network (NWK) Layer
Common Green Power (cGP) stub
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 2.3. Zigbee Stack Architecture

Green Power data exchanges are handled by a dedicated block or "stub" for the Zigbee Network (NWK) Layer and the Application Support Sub-layer (APS). Green Power has three elements within the Zigbee stack architecture:

- Common GP (cGP) stub
- Dedicated GP (dGP) stub
- Dedicated LPED (dLPED) stub

The following table describes each of the stubs in more detail.

Table 2.1. Green Power Stubs

Stub name	Description
Common GP (cGP)	Performs the basic functions shared by Low Power EndPoint (LPED) and GP. It performs just enough processing to pass application data frames to the Medium Access Control (MAC) layer for transmission and to pass the 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 Points (SAPs)—CGP-SAP and CZLPED-SAP.
Dedicated GP (dGP)	Performs just enough processing to pass application data frames to the cGP stub for transmission and to pass GPD commands from the cGP stub to the Green Power cluster on the Green Power EndPoint on receipt. The dGP stub is accessible to the higher layers through a special SAP—GP-SAP—which is parallel to the normal Application Support Sublayer Data Entity (APSDE)-SAP. The dGP communication architecture does not support simultaneous execution by multiple application entities. A Zigbee router is assumed to have only one proxy application entity (Green Power EndPoint) that will use the GP communication mechanism.
Dedicated LPED (dLPED)	This stub, as well as the corresponding LPED-SAPs, are out of scope of this User's Guide and will be defined separately by the Low Power End Device Task

2.2 Green Power Devices

GPDs are ultra-low power and even battery-less devices that can utilize this communications method to send messages to Zigbee devices. GPDs can only send and receive GPDFs. GPDs can never be a part of the Zigbee network because they have a different frame format. (See Section 4.1.1 Frame Format for more information.) Some typical GPDs are shown in the following figure.

Figure 2.4. Typical Green Power Devices

2.3 Green Power Infrastructure Devices

ZGP devices that operate within a normal Zigbee network are called *infrastructure devices*. These devices can handle GPDFs in some way. There are two general types of infrastructure devices:

- Green Power Proxy (GPP)
- Green Power Sink (GPS)

The GP specification defines the entire set of features for each of these devices. Some of these features are optional and some are mandatory, called *Basic* by Zigbee. As a result, there is a set of sub-names: Green Power Proxy Basic (GPPB), Green Power Sink Basic (GPSB) and Green Power Combo Basic (GPCB + GPSB) that implements the functionality for both Proxy and Sink within a single device.

The following figure illustrates two GP infrastructure devices.

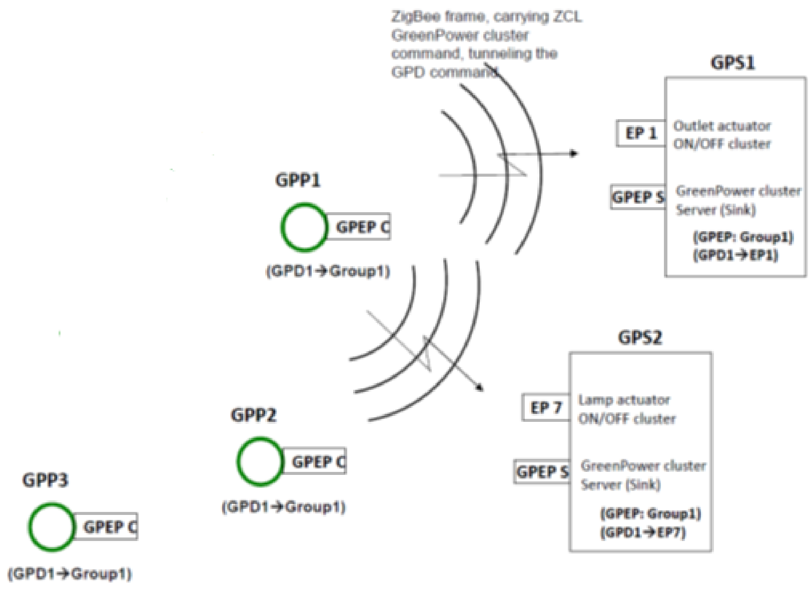

ZigBee frame, carrying ZCL GreenPower cluster command, tunneling the GPD command
GPS1
EP 1
Outlet actuator ON/OFF cluster
GPEP S
GreenPower cluster Server (Sink)
(GPEP: Group1)
(GPD1→EP1)
GPP1
GPEP C
(GPD1→Group1)
GPS2
EP 7
Lamp actuator ON/OFF cluster
GPEP S
GreenPower cluster Server (Sink)
(GPEP: Group1)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

Figure 2.5. Green Power Device Types

- Green Power Proxy (GPP)
  - Receives and forwards the GPDF wrapped in a ZCL command over the Zigbee network. This is sometimes called *tunneling* the GPD command.
  - Receives the GPDF, hence a Server for GPDF frames.
  - Sends the ZCL tunnel commands to the GPS. This is a client role of GP Cluster in ZCL.
- Green Power Sink (GPS)
  - Receives the tunneled commands and executes them.
  - Receives the ZCL tunnel commands from the GPP. This is a server role of GP Cluster in ZCL.

  Note: Even though it is possible to implement a standalone GPS device with EmberZNet PRO with the limitation of in-range direct communication with a GPD, a Green Power Combo (GPC) is preferred over a standalone GPS. The reason is that standalone sinks are rare because the Z3 specification requires all the routers to be Proxy Basic at minimum. The following sections of this User Guide discuss how to implement a GP Combo (GPC) rather than a GPS.

3. Green Power Commissioning

Commissioning is the process of adding a GPD to the Zigbee network so it can perform application functionality. This process establishes the GPD in the Proxy Table and in the Sink Table. If there is an infrastructure device (Sink or Combo) that has matching functionality with the GPD, they can be commissioned as a pair and work together.

The GPD sends a series of GP commands to the listening GP infrastructure device (Sink or Combo) which adds it as a commissioned node for further data transactions. In this command exchange, the GPD and the GP infrastructure device agree on the functionality that matches and the security requirements for additional command/data transactions.

There are two types of commissioning process based on the capabilities and design of the GPD:

- Unidirectional Commissioning: The GPD only sends the commissioning commands and does not receive any data back from the Zigbee network. In this type of commissioning, the application must predefine the Zigbee network channel and security requirements.
- Bidirectional Commissioning: The GPD sends and receives data through a series of commands, called *commissioning commands*. With bidirectional commissioning, the GPD can find the Zigbee network channel and negotiate the security level key. This process consumes more energy because it includes both transmission and receive steps.

3.1 Unidirectional Commissioning

The following figure illustrates how unidirectional commissioning works.

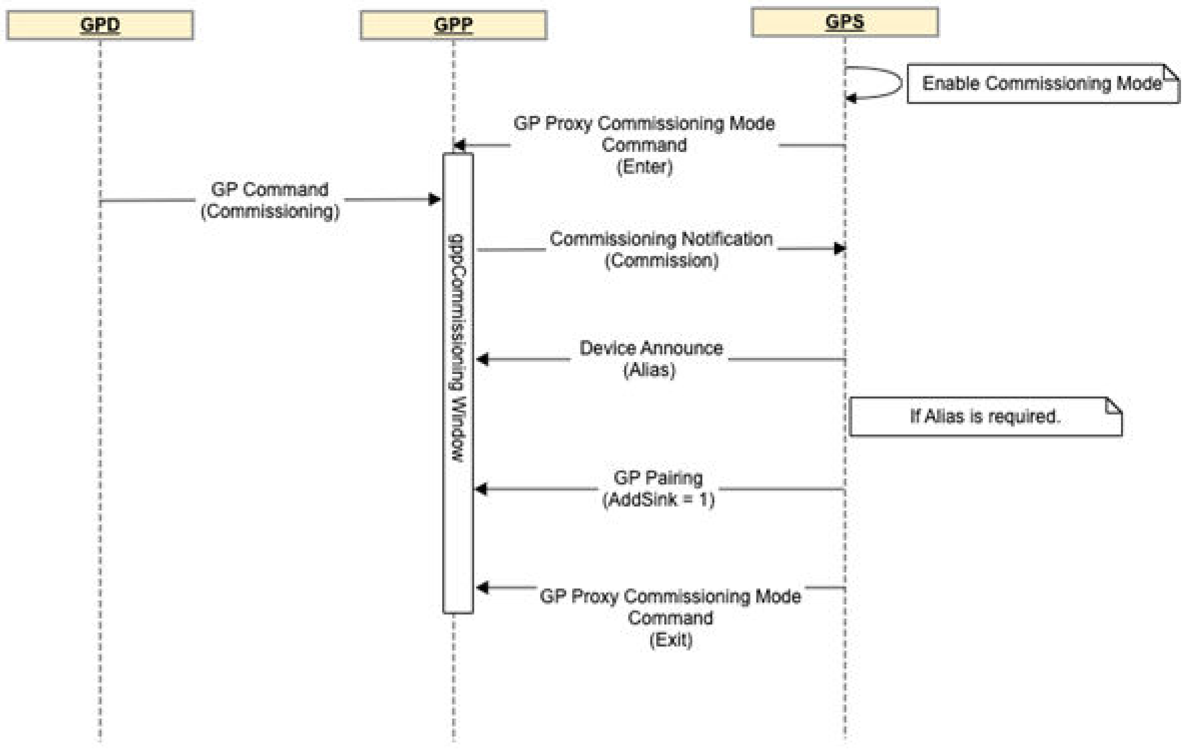

GPD
GPP
GPS
Enable Commissioning Mode
GP Proxy Commissioning Mode Command (Enter)
GP Command (Commissioning)
gppCommissioning Window
Commissioning Notification (Commission)
Device Announce (Alias)
If Alias is required.
GP Pairing (AddSink = 1)
GP Proxy Commissioning Mode Command (Exit)

Figure 3.1. Unidirectional Commissioning

- **GP Proxy Commissioning Mode (Enter):** The GPP will receive the GP Proxy Commissioning Mode from the GPS and start the GPP Commissioning Window timeout, which is the timeout for the GPP to exit Commissioning mode in case there is no pairing complete or on an exit command from the GPS. In commissioning mode the GPP will only accept GP Proxy Commissioning mode commands from the original GPS.
- **GP Command (Commissioning):** Once the GPS and GPP are in commissioning mode, an action is performed on the Green Power Device (GPD) to send a Commissioning GPDF which will be received by the proxy. The proxy will check if this device is already in the proxy table. If not, an active entry with default values will be created.
- **Commissioning Notification (Commission):** The GPP broadcasts a commissioning notification to the GPS. The GPS checks a functionality match, if security-related fields are supported (Key Type, Security Level, etc.). Once all the checks pass, the GPD capability is stored and the GPS proceeds to finalize commissioning by adding the GPD to the Sink Table and broadcasting a GP Pairing. If it is required, the GPS will assign the GPD an Assigned Alias. You have the option to define the Assigned Alias fo the GPD by using

  `bool sl_zigbee_af_green_power_server_update_alias_cb(sl_zigbee_gp_address_t *gpdAddr, uint16_t *alias)`: This function is called by the Green Power Server plugin during commissioning to updated the alias information from the user.

Once commissioning is finalized the GPS may send a GP Proxy Commissioning Mode Command (Exit) to have the GPP exit commissioning mode, or just wait for the GPP Commissioning Window timeout.

3.2 Bidirectional Commissioning

The following figure illustrates how bidirectional commissioning works.

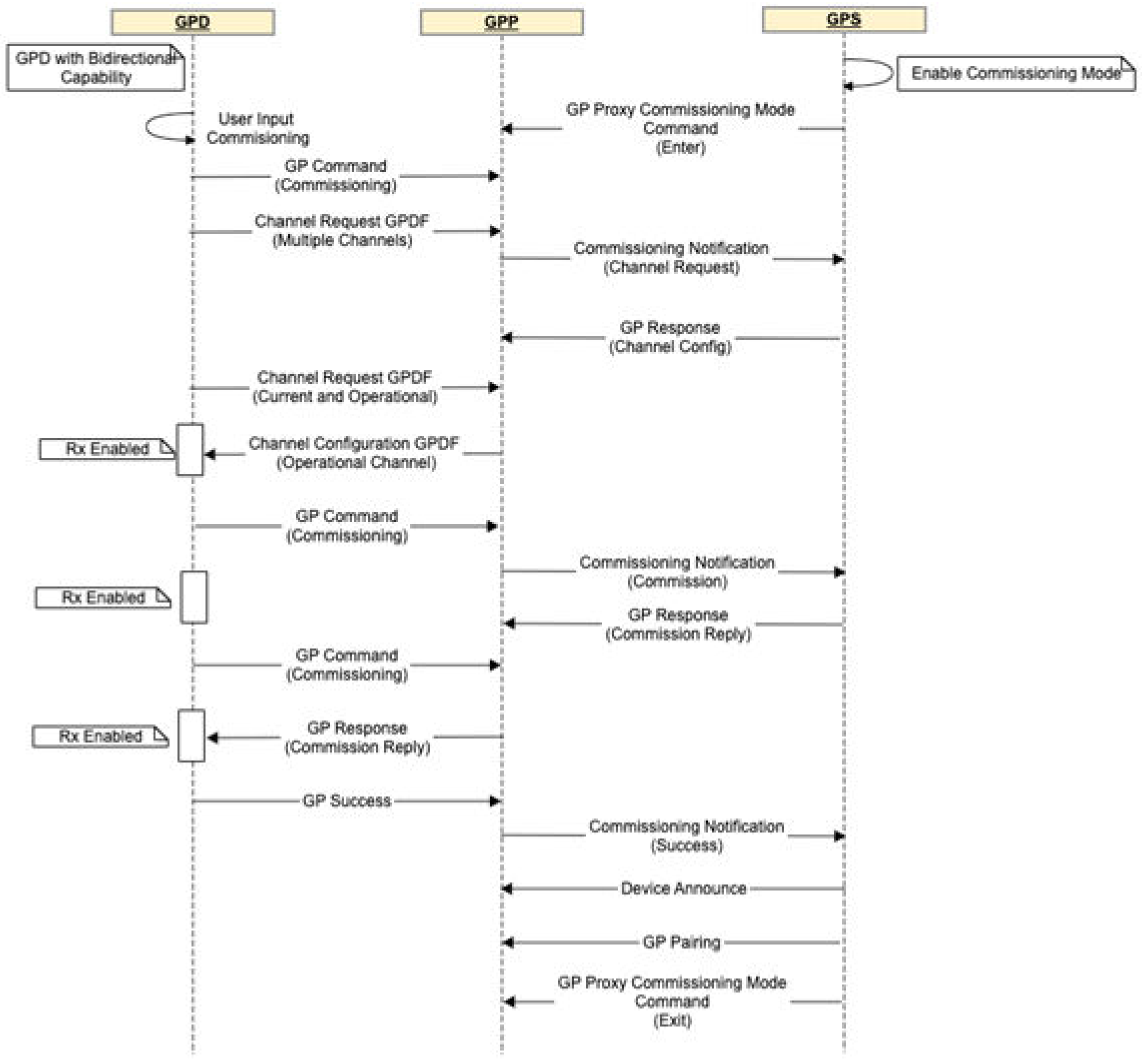

GPD
GPP
GPS
GPD with Bidirectional Capability
Enable Commissioning Mode
User Input Commisioning
GP Proxy Commissioning Mode Command (Enter)
GP Command (Commissioning)
Channel Request GPDF (Multiple Channels)
Commissioning Notification (Channel Request)
GP Response (Channel Config)
Channel Request GPDF (Current and Operational)
Rx Enabled
Channel Configuration GPDF (Operational Channel)
GP Command (Commissioning)
Commissioning Notification (Commission)
Rx Enabled
GP Response (Commission Reply)
GP Command (Commissioning)
Rx Enabled
GP Response (Commission Reply)
GP Success
Commissioning Notification (Success)
Device Announce
GP Pairing
GP Proxy Commissioning Mode Command (Exit)

Figure 3.2. Bidirectional Commissioning

Upon the expiry of any commissioning timer, such as the server commissioning window expiry, generic switch commissioning or multi-sensor commissioning timer, the Green Power Server plugin will call:

```
void sl_zigbee_af_green_power_server_commissioning_timeout_cb(uint8_t commissioningTimeoutType,
                                                               uint8_t numberOfEndpoints,
                                                               uint8_t * endpoints)
```

During commissioning if you need to check the pairing status of a commissioning gpd you can use :

```
void sl_zigbee_af_green_power_server_pairing_status_cb(EmberSinkPairingStatus status,
                                                       EmberCommissioningGpd *commissioningGpd))
```

This callback can be monitored to get information in case a GPD Commissioning that has started ended up in success or failure. If a commissioning GPDF is dropped by the GPS this callback will not provide any information regarding the GPDF.

4. Green Power Operational Model

A GPD is said to be *operational* when it sends application control commands such as on/off. Typically, such operational commands are useful after the GPD is commissioned to a Zigbee network after the commissioning process. Like commissioning, a GPD can have two operational modes.

- Unidirectional Operation: Sends commands to the Zigbee network and requires a minimum energy budget.
- Bidirectional Operation: Sends commands and receives a response back—for example, reading an attribute from the network. Because this mode sends data to the Zigbee network and receives data back from the network, it uses more energy.

4.1 Green Power Command Transport

The following figure illustrates a high-level typical GP operational command transport with four GP devices.

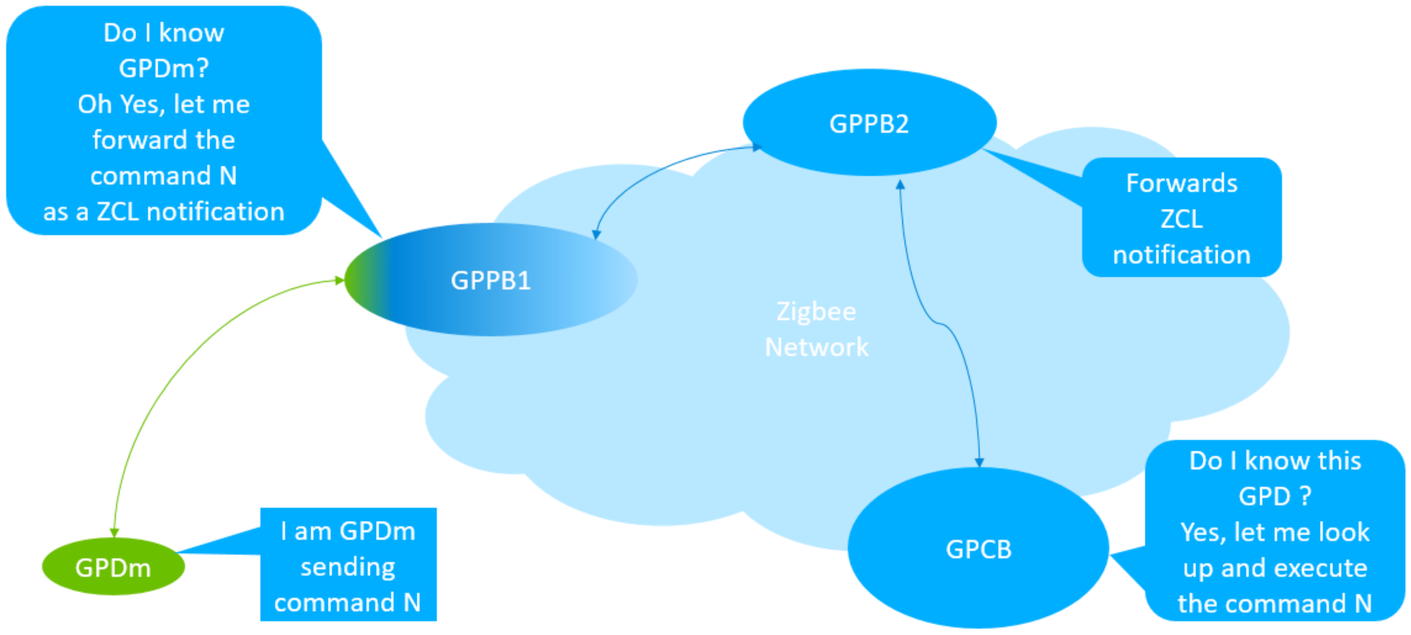

Do I know GPDm? Oh Yes, let me forward the command N as a ZCL notification
GPPB2
Forwards ZCL notification
GPPB1
Zigbee Network
Do I know this GPD ? Yes, let me look up and execute the command N
GPCB
GPDm
I am GPDm sending command N

Figure 4.1. Green Power Command Transport

These are the general steps in the Green Power command transport in the above figure.

1. GPDm (a GPD with Source Id = m) sends a command N (a GPDF frame) to GPPB1.
2. GPPB1 receives the command N.
3. GPPB1 looks up GPDm in its proxy table to determine whether GPDm has an entry paired with a Sink. If it does, GPPB1 forwards command N as a Zigbee Cluster Library (ZCL) notification to GPPB2. If it does not recognize GPDm after the proxy table lookup, GPPB1 drops the packet.
4. Once GPPB2 receives the notification, it forwards the command further as a ZCL command, as it would do for any other ZCL commands in the Zigbee network. (Any router in the Zigbee network would do the same because in Z3, all routers are GPPBs.)
5. The GPCB receives the ZCL notification and processes it by looking up its Sink table to obtain the required information. If all the information is correct and the GPDm is paired to this GPCB, the GPCB processes the command as follows:

   1. Looks up command N in its translation table to attempt to translate it.

   2. Translates command N to an equivalent Zigbee command (which includes interpreting the command payload) and finds one or more paired application endpoints (which are in the translation table). Refer to the translation table in Table 4.1 Default Translation Table No Payload on page 11 for the Zigbee Commands.

   3. Pushes the Zigbee packet to the identified application endpoint on the node.

4.1.1 Frame Format

The following figures illustrate the standard format of a Zigbee frame and the format of a generic GP frame. The size difference is because the entire GP frame must fit into the ZCL payload (or addressing).

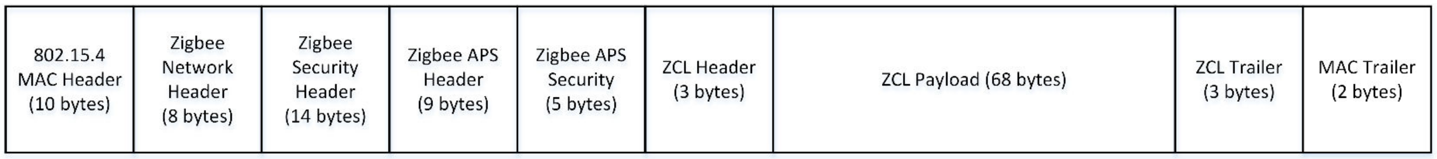

802.15.4 MAC Header (10 bytes)
Zigbee Network Header (8 bytes)
Zigbee Security Header (14 bytes)
Zigbee APS Header (9 bytes)
Zigbee APS Security (5 bytes)
ZCL Header (3 bytes)
ZCL Payload (68 bytes)
ZCL Trailer (3 bytes)
MAC Trailer (2 bytes)

Figure 4.2. Zigbee Frame Format

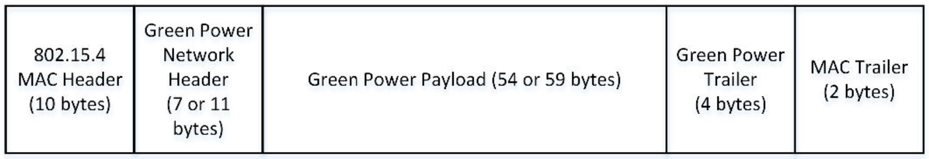

802.15.4 MAC Header (10 bytes)
Green Power Network Header (7 or 11 bytes)
Green Power Payload (54 or 59 bytes)
Green Power Trailer (4 bytes)
MAC Trailer (2 bytes)

Figure 4.3. Green Power Frame Format

Most of the information contained in the Zigbee NWK header and all the information in the APS headers is not relevant for GP operation. As a result, the GP frame contains a modified NWK header and no APS header, followed by a dedicated application payload. GP frames are compact and shorter, but they contain all the necessary information required for addressing and security.

4.1.2 Green Power Device Tables

The table sizes (sink and proxy tables) are configured using the configurations in *green-power-translation-table-config.h*. The translation table is to support configurable translation for compact attribute reporting and multi-sensor **CJO multi-sensor operation?**. Compact attribute reporting is supported in the EmberZnet stack as part of supporting the translation table.

To support the Green Power feature, the following tables are maintained on the sink and/or proxy nodes:

- **Proxy Table**: This table holds pairing information about the paired GPDs and the corresponding Sink. An entry to the proxy table is automatically activated during the commission process. The application can access the proxy table entries with the following APIs:
  - `emberGpProxyTableLookup(EmberGpAddress *addr)`: Look up a specific GPD source id in the Proxy table.
  - `emberAfPluginGreenPowerClientClearProxyTable()`: Clear the Proxy table of all stored GPD pairings.
- **Sink table:** This table holds information about local bindings between a particular GPD and the target's local endpoints in the GPS. An entry to the sink table is automatically activated during the commission process. The application can access sink table entries with the following APIs:
  - `uint16_t emberAfFillCommandGreenPowerClusterGpSinkTableRequestSmart (uint8_t options, uint32_t gpdSrcId, uint8_t *gpdIeee, uint8_t gpdEndpoint, uint8_t index)`: This function prepares an application framework ZCL command buffer for the GP sink table request command with supplied arguments and returns the buffer length.
  - `bool emberAfPluginGreenPowerServerSinkTableAccessNotificationCallback(void* data, EmberAfGpServerSinkTableAccessType accessType)`: This function is called by the green power server plugin to notify the application about a GPD addition or removal by the GPS to the Sink table.
- **Translation table** : Holds translations for GPD command into a command from the Zigbee application profile supported on the node:
  - The commands from the source (GPD) node do not come from a standard Zigbee command set. A sink node for GP commands interprets the commands received from the GPD and performs the required action. The sink node must translate the received command into a command from the Zigbee Application profile supported on the node (for exmple, Home Automation).
  - A Translation table entry is derived from the Default or Customized table.
    - **The Default Table**: Stores the predefined translations as per GP specification. By default, the GPD Command Translation Table (Table 1) contains the generic translations (mapping the GPD commands to their ZCL equivalents).
    - **Customized Table**: Contains specific command translations defined for a given GPD device. Creating a customized table is outside the scope of this User Guide.
  - If both default and customized translation are applicable to a particular GPD command, the customized translation supersedes the default translation.
  - The translation table is created during commissioning and later used to the interpret the command received from GPD.

Table 4.1. Default Translation Table No Payload

GPD Commands		Mapping to Zigbee		
CommandID	**Command Name**	**Corresponding Cluster-ID**	**CommandID**	**Command Payload**
0x00	Identify	Identify	Identify	0x003xc
0x01 - 0x0F	Reserved			
0x10 - 0x17	Recall Scene 0-7	Scenes	Recall Scene	GroupID, SceneID = 0 - 7
0x18-0x1F	Store Scene 0	Scenes	Store Scene	GroupID, SceneID = 0
0x20	Off	On/Off	Off	N/A
0x21	On	On/Off	On	N/A
0x22	Toggle	On/Off	Toggle	N/A
0x23	Release	-		
0x24 - 0x2F	Reserved			
0x30 - 0x33	Defined in Table 4.2 GPDF Commands with Payload on page 12 sent by GPD			
0x34	Level Control/Stop	Level Control	Stop	N/A
0x35 - 0x38	Defined in Table 4.2 GPDF Commands with Payload on page 12 sent by GPD			

GPD Commands		Mapping to Zigbee		
CommandID	**Command Name**	**Corresponding Cluster-ID**	**CommandID**	**Command Payload**
0x39-0x3F	Reserved			
0x40	Move Hue Stop	Color Control	Move Hue	Stop
0x41 - 0x44	Defined in Table 4.2 GPDF Commands with Payload on page 12 sent by GPD			
0x45	Move Saturation Stop	Color Control	Move Saturation	Stop
0x46 - 0x4B	Defined in Table 4.2 GPDF Commands with Payload on page 12 sent by GPD			
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 - 0x6A	Defined in Table 4.2 GPDF Commands with Payload on page 12 sent by GPD			
0x6B - 0x6F	Reserved			
0x70 - 0x9F	Reserved			
0xA0 - 0xE0	Defined in Table 4.2 GPDF Commands with Payload on page 12 sent by GPD			
0xE1	Decommissioning	N/A		
0xE2	Success	N/A		
0xE3	Defined in Table 4.2 GPDF Commands with Payload on page 12 sent by GPD			
0xE4 - 0xEF	Defined in Table 4.2 GPDF Commands with Payload on page 12 sent by GPD			

Table 4.2. GPDF Commands with Payload

GPD Command		Mapping to Zigbee	
CommandID	**Command Name**	**ClusterID**	**Command Name**
0x30	Move Up	Level Control	Move Up
0x31	Move Down	Level Control	Move Down
0x32	Step Up	Level Control	Step Up
0x33	Step Down	Level Control	Step Down
0x35	Move Up (with On/Off)	Level Control	Move Up (with On/Off)

GPD Command		Mapping to Zigbee	
CommandID	**Command Name**	**ClusterID**	**Command Name**
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
0xA1	Manufacturer Specific Attribute Reporting	Copied from the triggering GPD command	ZCL Report Attributes Command
0xA2	Multi-Cluster Reporting	Copied from the triggering GPD command	ZCL Report Attributes Command
0xA3	Manufacturer Specific Multi-Cluster Reporting	Copied from the triggering GPD command	ZCL Report Attributes Command
0xA4	Request Attributes	Copied from the triggering GPD command	ZCL Report Attributes Command
0xA5	Read Attribute Response	Copied from the triggering GPD command	ZCL Read Attributes Response command
0xA6	ZCL Tunneling	Copied from the triggering GPD command	Copied from the triggering GPD command
0xA7	Reserved		
0xA87	Compact Attribute Reporting	Derived from the triggering GPD command, using the information sent during commissioning	ZCL Report attributes command
0xA897-0xAE	Reserved		
0xAF	GPD sensor commands (0xA0 – 0xA3)	Copied from the triggering GPD command	ZCL Report attributes command
0xB0 - 0xBF	Manufacturer-defined GPD commands (payload is manufacturer-specific)		
0xC0 - 0xDF	Reserved		
0xE0	Commissioning	N/A	
0xE3	Channel Request	N/A	
0xE4	Application Description	N/A	
0xE54 - 0xEF	Reserved		

5. Silicon Labs Green Power within Zigbee EmberZNet v6.x and Lower

There are three Silicon Labs Green Power devices within EmberZNet PRO:

- Green Power Device
- Green Power Proxy Basic
- Green Power Combo Basic

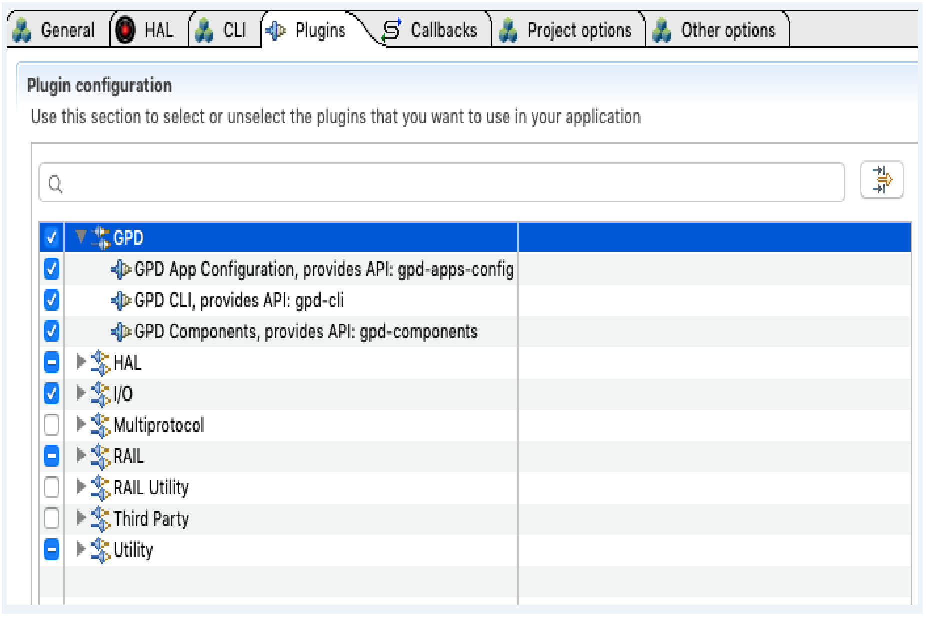
The following sections describe some of the main features, plugins, and callbacks for each device type.

5.1 Green Power Device

A device that is outside a Zigbee network and can send commands to the Zigbee network via a GPP.

5.1.1 Plugins

These are the plugins related to a GPD application and its components available in the GPD framework.



Plugin configuration
Use this section to select or unselect the plugins that you want to use in your application

- [✓] GPD
  - [✓] GPD App Configuration, provides API: gpd-apps-config
  - [✓] GPD CLI, provides API: gpd-cli
  - [✓] GPD Components, provides API: gpd-components
- [–] HAL
- [✓] I/O
- [ ] Multiprotocol
- [–] RAIL
- [ ] RAIL Utility
- [ ] Third Party
- [–] Utility

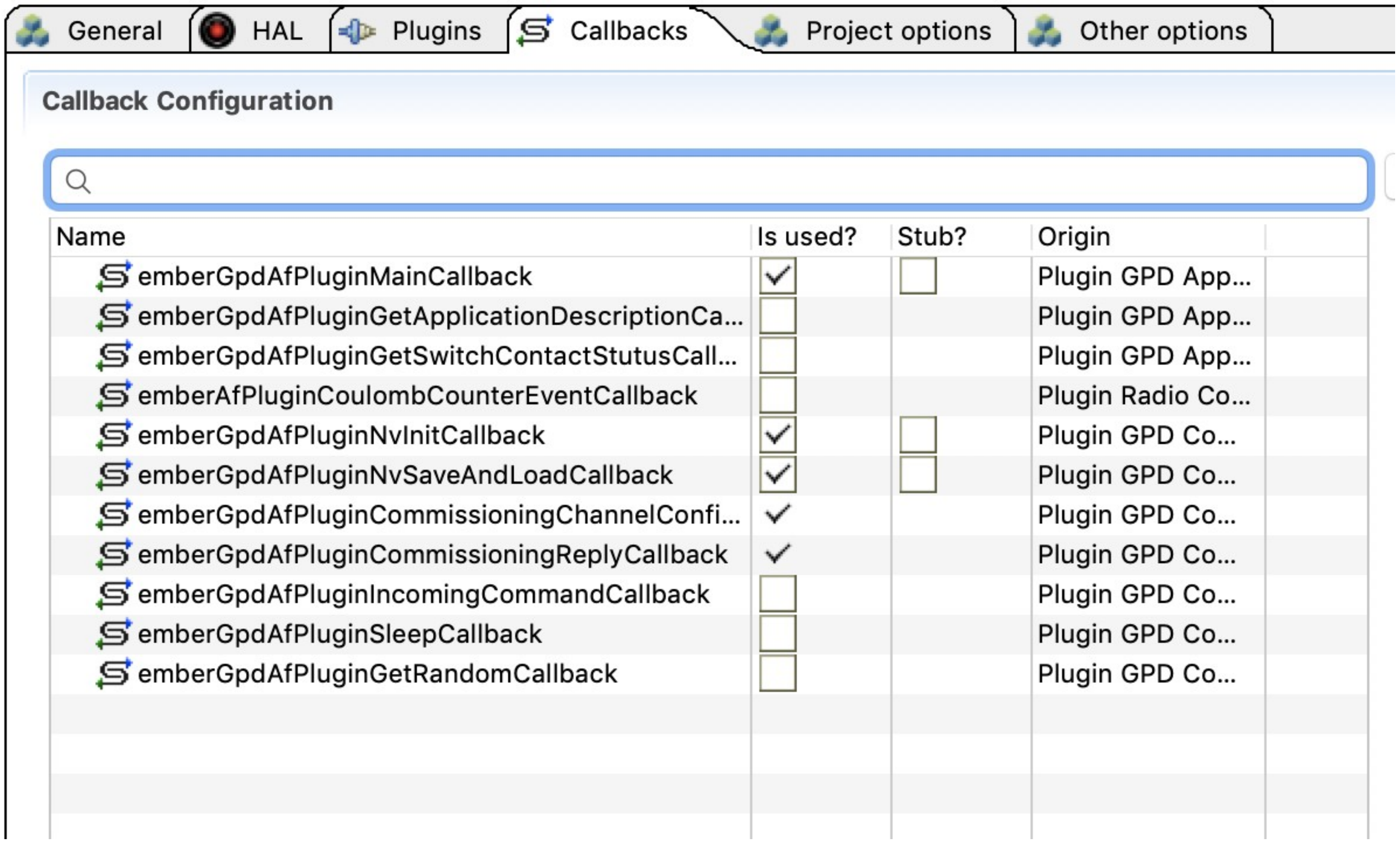
Figure 5.1. Green Power Device Plugins

The GPD framework provides functional modules in the form of the following plugins:

- GDP
  - GPD App Configuration: Configures application capability and device parameters. It also consists of a set of modules that provide application-level functions such as main, node configuration, and commissioning.
  - GPD CLI: Provides a basic set of Command Line Interface (CLI) commands for development and testing.
  - GPD Components: Includes a set of modules to provide the following capabilities:
    - Wrapper interface to RAIL layer
    - Sending and receiving bidirectional GPDFs
    - GPDF packet formation and parsing
    - GPDF security tagging and validation wrappers for the mbed TLS plugin
    - Nonvolatile memory wrapper functions
- HAL (Hardware Abstraction Layer): HAL Library includes a set of low-layer peripheral and board support modules.
- RAIL (Radio Abstraction Interface Layer): Includes the RAIL library.
- Utility: Provides mbedTLS functionality.

5.1.2 Callbacks

The following figure illustrates the set of callbacks available for user interaction that provide information and accept inputs.



Callback Configuration

Name	Is used?	Stub?	Origin
emberGpdAfPluginMainCallback	✓	☐	Plugin GPD App...
emberGpdAfPluginGetApplicationDescriptionCa...	☐		Plugin GPD App...
emberGpdAfPluginGetSwitchContactStutusCall...	☐		Plugin GPD App...
emberAfPluginCoulombCounterEventCallback	☐		Plugin Radio Co...
emberGpdAfPluginNvInitCallback	✓	☐	Plugin GPD Co...
emberGpdAfPluginNvSaveAndLoadCallback	✓	☐	Plugin GPD Co...
emberGpdAfPluginCommissioningChannelConfi...	✓		Plugin GPD Co...
emberGpdAfPluginCommissioningReplyCallback	✓		Plugin GPD Co...
emberGpdAfPluginIncomingCommandCallback	☐		Plugin GPD Co...
emberGpdAfPluginSleepCallback	☐		Plugin GPD Co...
emberGpdAfPluginGetRandomCallback	☐		Plugin GPD Co...

Figure 5.2. Green Power Device Callbacks

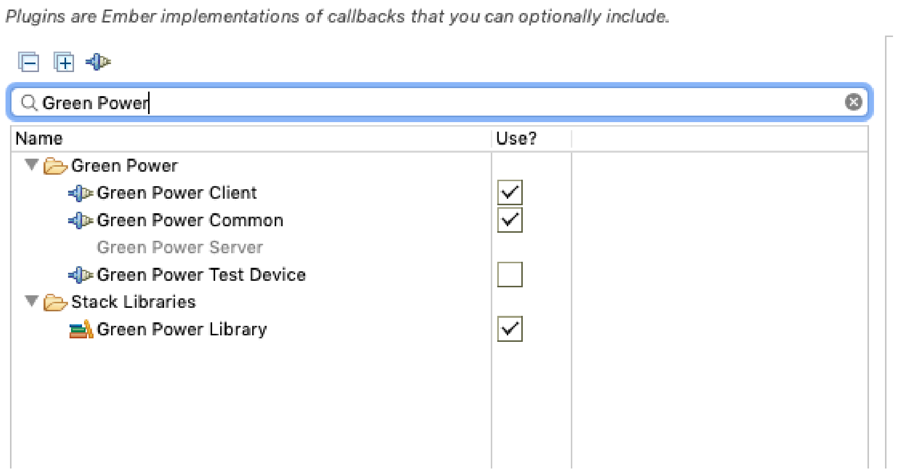
5.2 Green Power Proxy Basic

Green Power Proxy Basic has the following key features:

- Receives Green Power Frames.
- Converts Green Power Frames to ZCL commands.
- Is involved in commissioning new devices.

5.2.1 Plugins

Plugins implement the functionality for the mandatory incoming commands. On the Plugin tab, ensure the following plugins are enabled.



Plugins are Ember implementations of callbacks that you can optionally include.

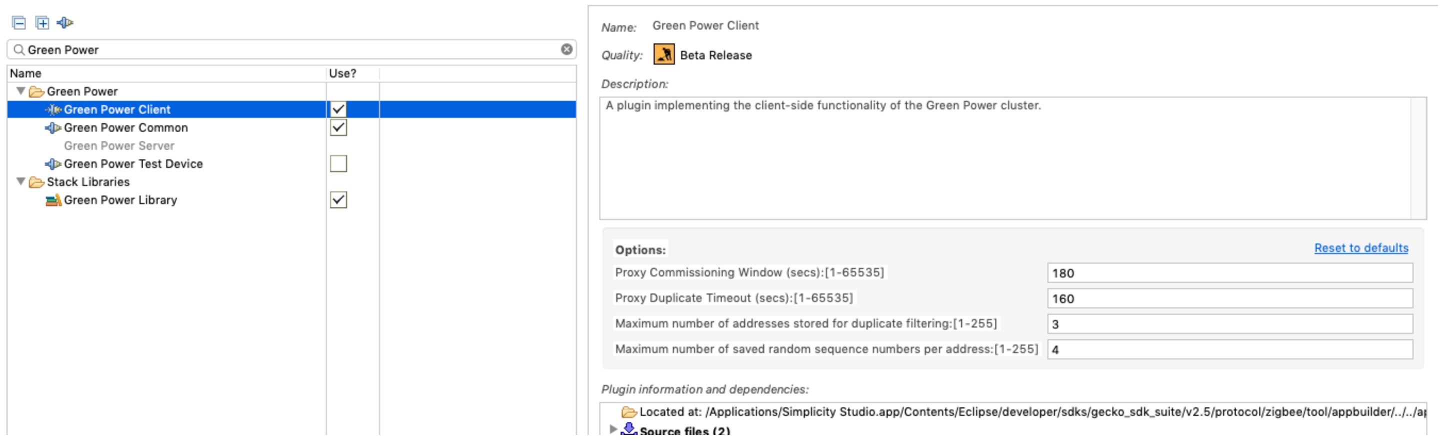
Search: Green Power

Name	Use?
Green Power	
Green Power Client	✓
Green Power Common	✓
Green Power Server	
Green Power Test Device	☐
Stack Libraries	
Green Power Library	✓

Figure 5.3. Green Power Proxy Basic Plugins

The Green Power Client plugin implements most of the Proxy functionality while the Green Power Common plugin provides some of the common functions and utility that is shared by both the Green Power Server and the Green Power Client.

When you enable a Green Power Client plugin, make sure to set the correct desired options as shown in the following figure.



Search: Green Power

Name	Use?
Green Power	
Green Power Client	✓
Green Power Common	✓
Green Power Server	
Green Power Test Device	☐
Stack Libraries	
Green Power Library	✓

Name: Green Power Client

Quality: Beta Release

Description:

A plugin implementing the client-side functionality of the Green Power cluster.

Options: Reset to defaults

Proxy Commissioning Window (secs):[1-65535]	180
Proxy Duplicate Timeout (secs):[1-65535]	160
Maximum number of addresses stored for duplicate filtering:[1-255]	3
Maximum number of saved random sequence numbers per address:[1-255]	4

Plugin information and dependencies:

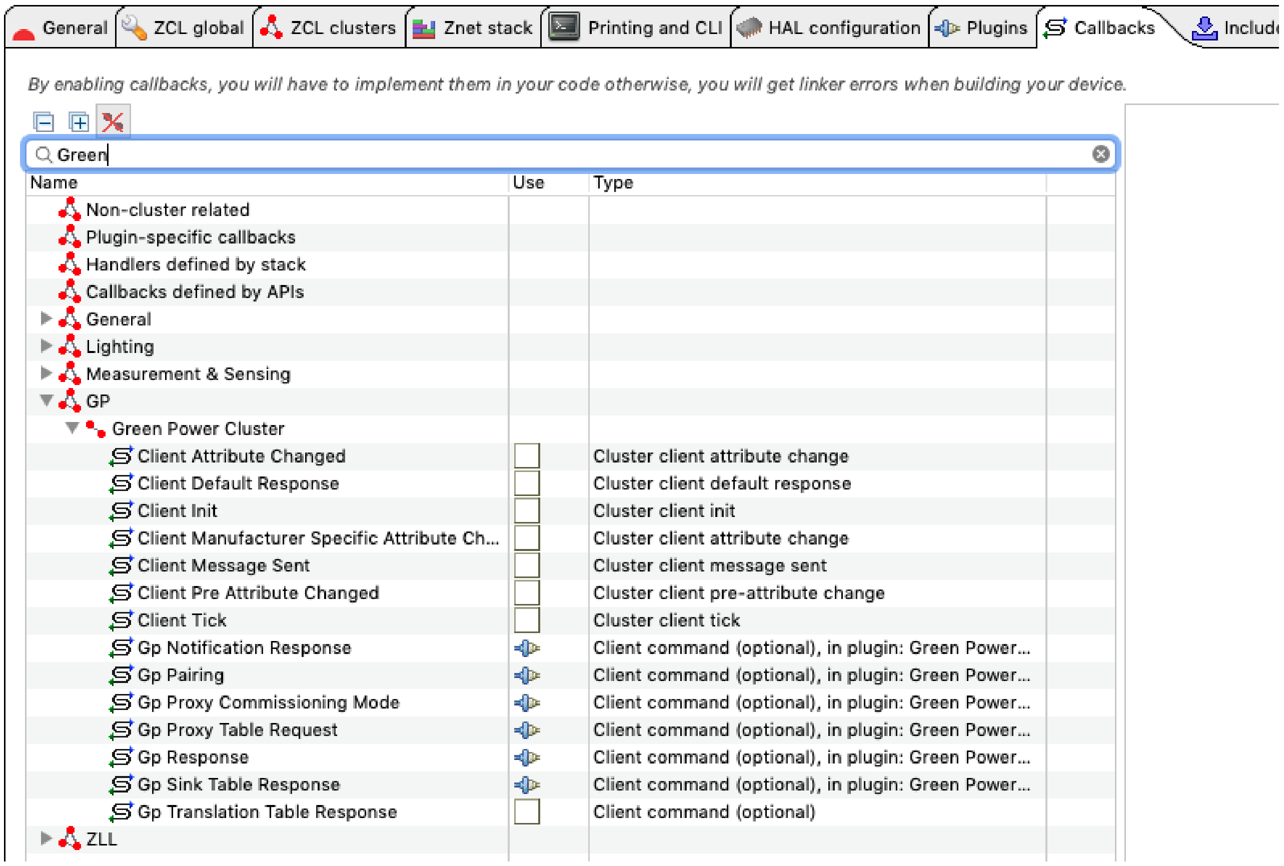
Located at: /Applications/Simplicity Studio.app/Contents/Eclipse/developer/sdks/gecko_sdk_suite/v2.5/protocol/zigbee/tool/appbuilder/../../a

Source files (2)

Figure 5.4. Green Power Client Plugin Options

5.2.2 Callbacks

The following figure illustrates the Green Power cluster incoming commands callbacks that are implemented by the Green Power Client as part of the Green Power Proxy Basic.



By enabling callbacks, you will have to implement them in your code otherwise, you will get linker errors when building your device.

Green

Name	Use	Type
Non-cluster related		
Plugin-specific callbacks		
Handlers defined by stack		
Callbacks defined by APIs		
▶ General		
▶ Lighting		
▶ Measurement & Sensing		
▼ GP		
▼ Green Power Cluster		
Client Attribute Changed	☐	Cluster client attribute change
Client Default Response	☐	Cluster client default response
Client Init	☐	Cluster client init
Client Manufacturer Specific Attribute Ch...	☐	Cluster client attribute change
Client Message Sent	☐	Cluster client message sent
Client Pre Attribute Changed	☐	Cluster client pre-attribute change
Client Tick	☐	Cluster client tick
Gp Notification Response		Client command (optional), in plugin: Green Power...
Gp Pairing		Client command (optional), in plugin: Green Power...
Gp Proxy Commissioning Mode		Client command (optional), in plugin: Green Power...
Gp Proxy Table Request		Client command (optional), in plugin: Green Power...
Gp Response		Client command (optional), in plugin: Green Power...
Gp Sink Table Response		Client command (optional), in plugin: Green Power...
Gp Translation Table Response	☐	Client command (optional)
▶ ZLL		

Figure 5.5. Green Power Proxy Client Callbacks

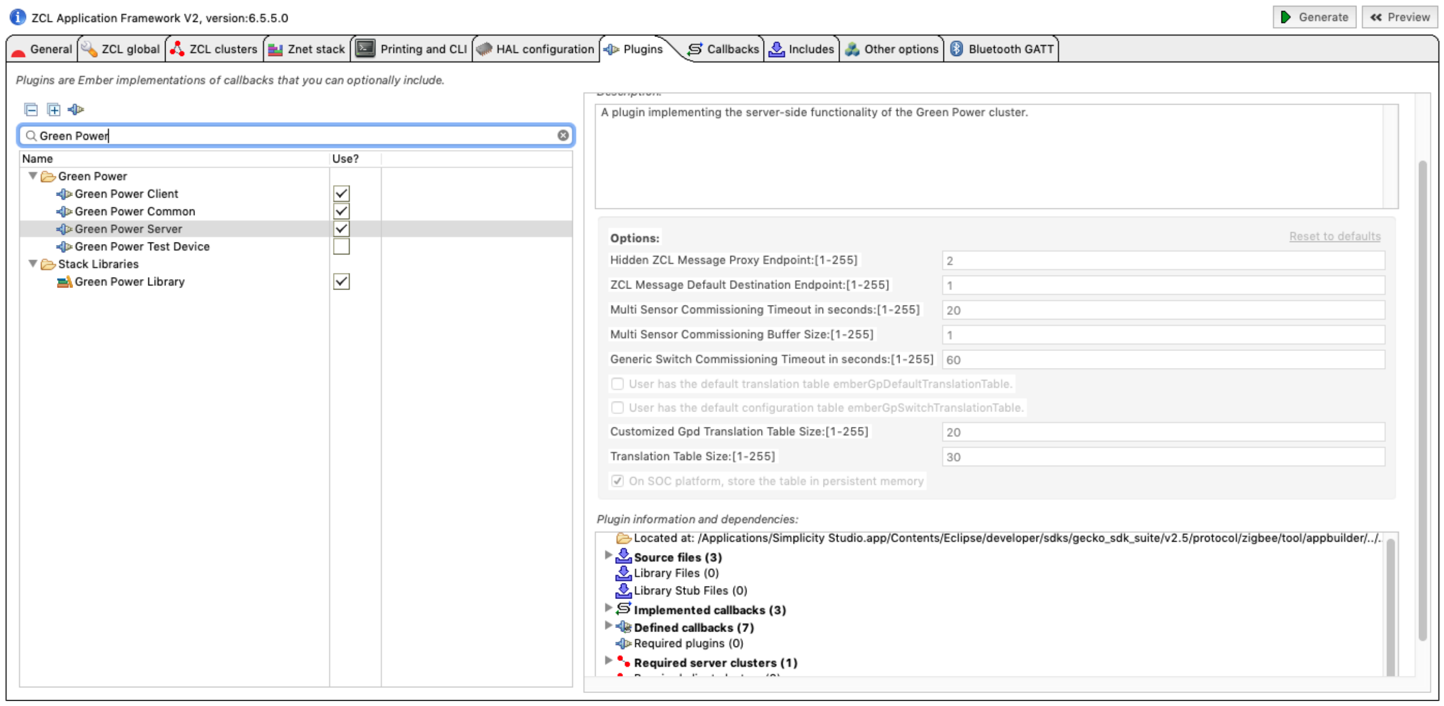
5.3 Green Power Combo Basic

Green Power Combo Basic has the following key features:

- Receives Green Power Frames.
- Converts Green Power Frames to ZCL commands.
- Commissions new devices.
- Is commanded by Green Power Devices.

5.3.1 Plugins

Plugins implement the functionality for the mandatory incoming commands for both the client and the server. On the Plugin tab, ensure the following plugins are enabled.


ZCL Application Framework V2, version:6.5.5.0

Generate | Preview

General | ZCL global | ZCL clusters | Znet stack | Printing and CLI | HAL configuration | Plugins | Callbacks | Includes | Other options | Bluetooth GATT

Plugins are Ember implementations of callbacks that you can optionally include.

Green Power

Name	Use?
Green Power	
Green Power Client	✓
Green Power Common	✓
Green Power Server	✓
Green Power Test Device	☐
Stack Libraries	
Green Power Library	✓

Description:

A plugin implementing the server-side functionality of the Green Power cluster.

Options: Reset to defaults

Hidden ZCL Message Proxy Endpoint:[1-255]	2
ZCL Message Default Destination Endpoint:[1-255]	1
Multi Sensor Commissioning Timeout in seconds:[1-255]	20
Multi Sensor Commissioning Buffer Size:[1-255]	1
Generic Switch Commissioning Timeout in seconds:[1-255]	60
☐ User has the default translation table emberGpDefaultTranslationTable.	
☐ User has the default configuration table emberGpSwitchTranslationTable.	
Customized Gpd Translation Table Size:[1-255]	20
Translation Table Size:[1-255]	30
✓ On SOC platform, store the table in persistent memory	

Plugin information and dependencies:

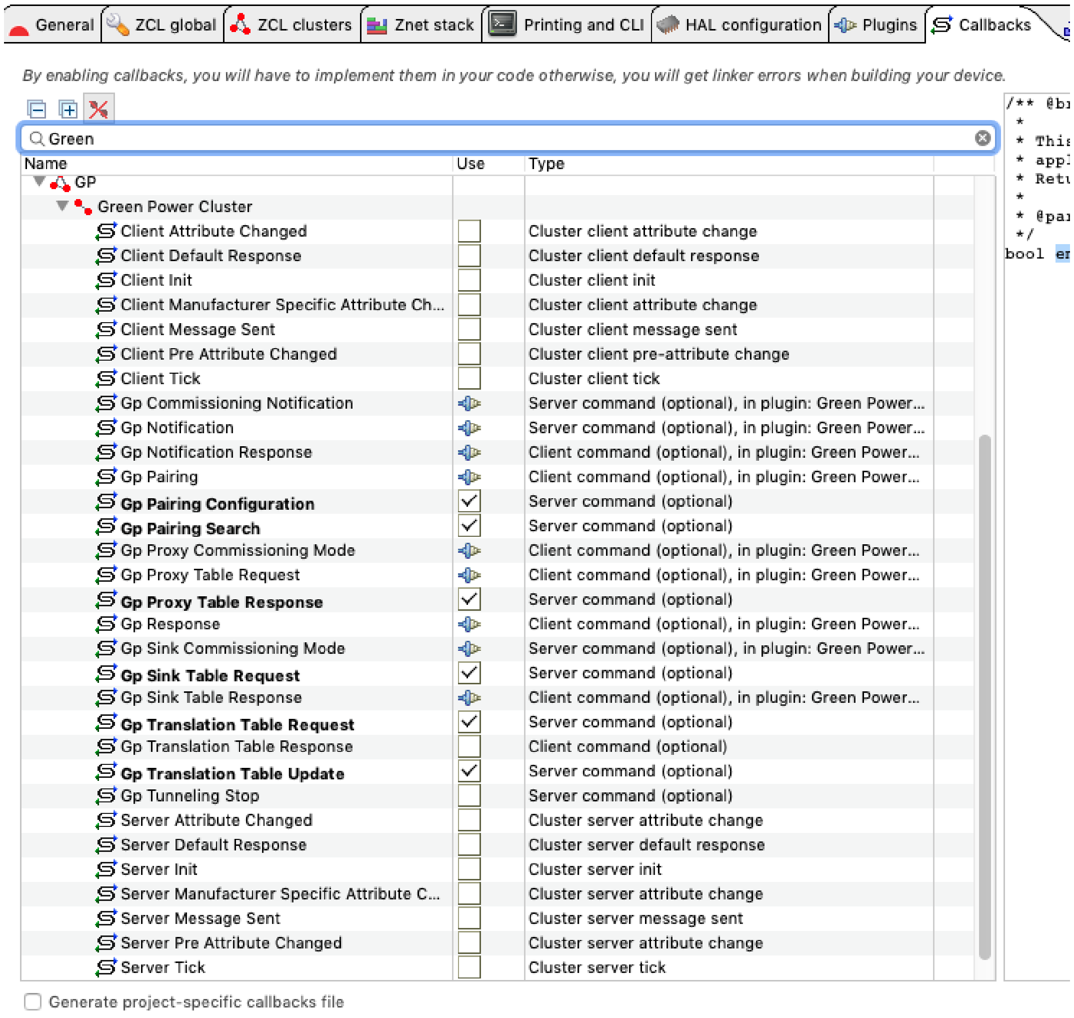
Located at: /Applications/Simplicity Studio.app/Contents/Eclipse/developer/sdks/gecko_sdk_suite/v2.5/protocol/zigbee/tool/appbuilder/../.
Source files (3)
Library Files (0)
Library Stub Files (0)
Implemented callbacks (3)
Defined callbacks (7)
Required plugins (0)
Required server clusters (1)

Figure 5.6. Green Power Sink Combo Plugins

Set **Hidden ZCL Message Proxy Endpoint** to one of the application endpoints implemented by the Green Power Combo Basic to allow the AF to send the operational command forwarding. Set **ZCL Message Default Destination Endpoint** to one of the application endpoints.

5.3.2 Callbacks

The following figure illustrates the Green Power Cluster incoming command callbacks that are implemented by the Green Power Client and the Green Power Server.



By enabling callbacks, you will have to implement them in your code otherwise, you will get linker errors when building your device.

Green

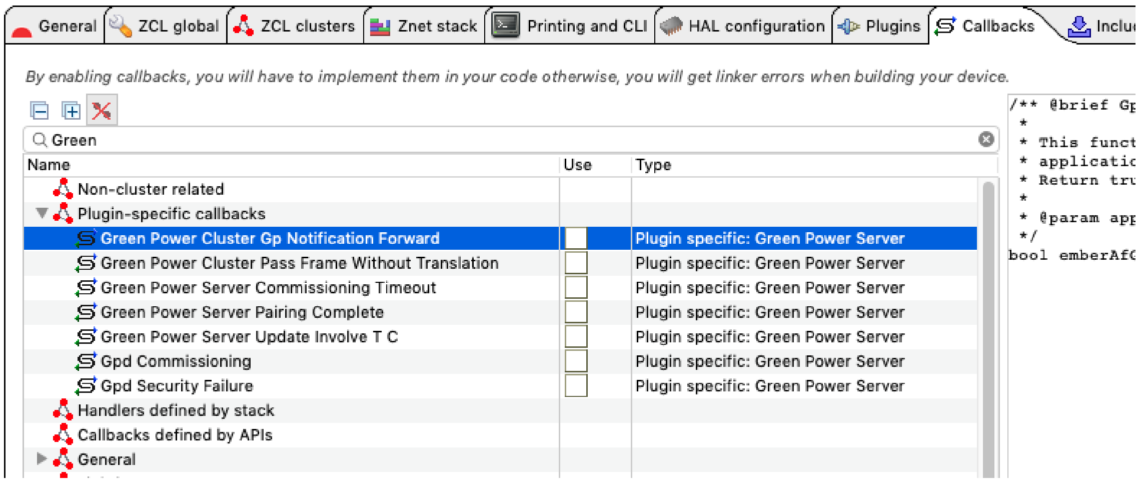
Name	Use	Type
GP		
Green Power Cluster		
Client Attribute Changed	☐	Cluster client attribute change
Client Default Response	☐	Cluster client default response
Client Init	☐	Cluster client init
Client Manufacturer Specific Attribute Ch...	☐	Cluster client attribute change
Client Message Sent	☐	Cluster client message sent
Client Pre Attribute Changed	☐	Cluster client pre-attribute change
Client Tick	☐	Cluster client tick
Gp Commissioning Notification		Server command (optional), in plugin: Green Power...
Gp Notification		Server command (optional), in plugin: Green Power...
Gp Notification Response		Client command (optional), in plugin: Green Power...
Gp Pairing		Client command (optional), in plugin: Green Power...
Gp Pairing Configuration	✓	Server command (optional)
Gp Pairing Search	✓	Server command (optional)
Gp Proxy Commissioning Mode		Client command (optional), in plugin: Green Power...
Gp Proxy Table Request		Client command (optional), in plugin: Green Power...
Gp Proxy Table Response	✓	Server command (optional)
Gp Response		Client command (optional), in plugin: Green Power...
Gp Sink Commissioning Mode		Server command (optional), in plugin: Green Power...
Gp Sink Table Request	✓	Server command (optional)
Gp Sink Table Response		Client command (optional), in plugin: Green Power...
Gp Translation Table Request	✓	Server command (optional)
Gp Translation Table Response	☐	Client command (optional)
Gp Translation Table Update	✓	Server command (optional)
Gp Tunneling Stop	☐	Server command (optional)
Server Attribute Changed	☐	Cluster server attribute change
Server Default Response	☐	Cluster server default response
Server Init	☐	Cluster server init
Server Manufacturer Specific Attribute C...	☐	Cluster server attribute change
Server Message Sent	☐	Cluster server message sent
Server Pre Attribute Changed	☐	Cluster server attribute change
Server Tick	☐	Cluster server tick

```
/** @br
 *
 * This
 * appl
 * Retu
 *
 * @par
 */
bool em
```

☐ Generate project-specific callbacks file

Figure 5.7. Green Power Cluster Command Callbacks

The following figure illustrates the set of callbacks that the Green Power Server uses as application callbacks.



By enabling callbacks, you will have to implement them in your code otherwise, you will get linker errors when building your device.

Green

Name	Use	Type
Non-cluster related		
Plugin-specific callbacks		
Green Power Cluster Gp Notification Forward	☐	Plugin specific: Green Power Server
Green Power Cluster Pass Frame Without Translation	☐	Plugin specific: Green Power Server
Green Power Server Commissioning Timeout	☐	Plugin specific: Green Power Server
Green Power Server Pairing Complete	☐	Plugin specific: Green Power Server
Green Power Server Update Involve T C	☐	Plugin specific: Green Power Server
Gpd Commissioning	☐	Plugin specific: Green Power Server
Gpd Security Failure	☐	Plugin specific: Green Power Server
Handlers defined by stack		
Callbacks defined by APIs		
General		

```
/** @brief Gp
 *
 * This funct
 * applicatic
 * Return tru
 *
 * @param app
 */
bool emberAfG
```

Figure 5.8. Green Power Server Application Callbacks

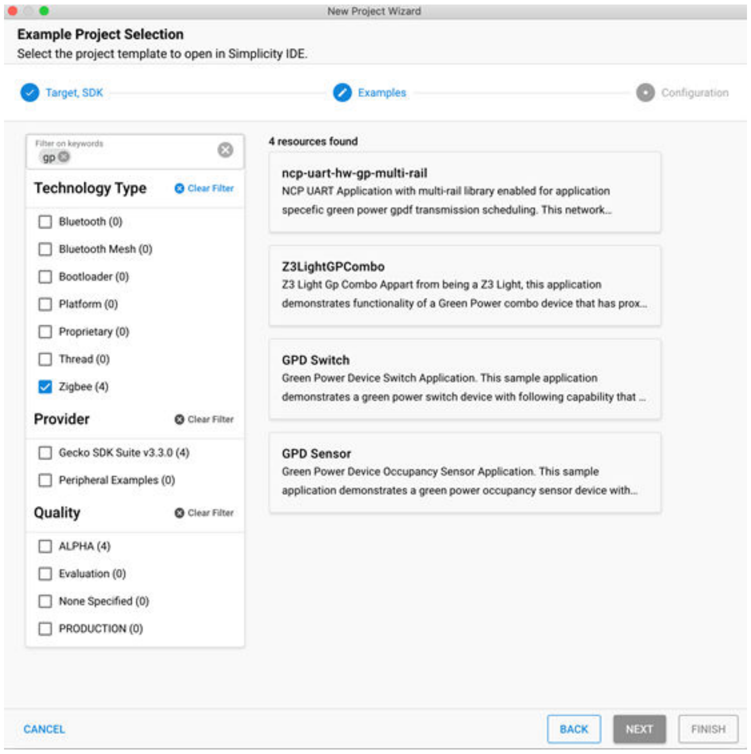
Note: Section 5.2 Green Power Proxy Basic and section 5.3 Green Power Combo Basic explained the implementation for a System-on-Chip (SoC) architecture. The Silicon Labs Zigbee application framework provides a Host-xNCP architecture for implementing a GPP or GPC. The plugins and callbacks remain the same and present on the host framework and the Green Power Library has the same settings as the above and remains on the xNCP.

6. Silicon Labs Green Power within Zigbee EmberZNet 8.x and Higher

Zigbee EmberZNet Software Development Kit (SDK) v8.x contains mainly API name changes compared to SDK v7.x. This chapter describes the Green Power features of the Zigbee EmberZNet 8.x SDK.

6.1 Examples - SoC

The same examples are provided as in SDK 6.10.x or earlier SDK version, however they can now be accessed from the File > New > Silicon Labs Project Wizard menu in Simplicity Studio 5. An easy way to access them is to filter for the "gp" keyword and Zigbee Technology Type.



Example Project Selection
Select the project template to open in Simplicity IDE.

Filter on keywords: gp

Technology Type — Clear Filter
- [ ] Bluetooth (0)
- [ ] Bluetooth Mesh (0)
- [ ] Bootloader (0)
- [ ] Platform (0)
- [ ] Proprietary (0)
- [ ] Thread (0)
- [✓] Zigbee (4)

Provider — Clear Filter
- [ ] Gecko SDK Suite v3.3.0 (4)
- [ ] Peripheral Examples (0)

Quality — Clear Filter
- [ ] ALPHA (4)
- [ ] Evaluation (0)
- [ ] None Specified (0)
- [ ] PRODUCTION (0)

4 resources found

ncp-uart-hw-gp-multi-rail
NCP UART Application with multi-rail library enabled for application specefic green power gpdf transmission scheduling. This network...

Z3LightGPCombo
Z3 Light Gp Combo Appart from being a Z3 Light, this application demonstrates functionality of a Green Power combo device that has prox...

GPD Switch
Green Power Device Switch Application. This sample application demonstrates a green power switch device with following capability that ...

GPD Sensor
Green Power Device Occupancy Sensor Application. This sample application demonstrates a green power occupancy sensor device with...

Figure 6.1. Green Power examples in the Simplicity Studio Project Wizard

The examples are:

- **Zigbee - GPD Sensor**: Green Power Device Occupancy Sensor application that demonstrates a Green Power occupancy sensor device. The Green Power Device Application Support component is pre-configured accordingly.
- **Zigbee - GPD Switch**: Green Power Device Switch application that demonstrates a Green Power switch device. The Green Power Device Application Support component is pre-configured accordingly.
- **Zigbee GP - Z3LightCombo**: The Z3Light example application extended with Green Power Combo (Proxy + Sink) functionality. Green Power-related components are added and pre-configured to support this functionality.
- **Zigbee - NCP UartHwGpMultiRail**: Zigbee - NCP UartHw example application extended with Green Power Multi-RAIL functionality.

For more information on the examples refer to the Project Details section of the Project Overview in each example.

6.2 Examples - NCP

In EmberZNet the basic functionality with Green Power endpoints is now located on NCP. This example shows how to run a Z3Gateway with a **Zigbee - NCP UartHwGpMultiRail rail sample** application and the necessary steps to pair a GPD Switch using the Green Power Adapter component. This sample application provides the following benefits:

- It allows the Zigbee Green Power to function entirely on the NCP with no intervention from the host in a Host-NCP setup and therefore reduces energy consumption by avoiding the need to wake up the Linux Host for Green Power packets.
- The GPPB and GP Sync is integrated into the NCP.
- It provides additional EZSP APIs added for the GP Sync functionality.

To start building your application follow these steps:

1. In Simplicity Studio create a new Z3Gateway Host Project. For information on how to create a Z3Gateway Host Project refer to *AN706: EZSP-UART Host Interfacing Guide*. Once the Z3Gateway project has been created, open the Project Configurator's Software Components tab:
    a. Under Zigbee -> Green Power uninstall the following components:
        - Green Power Client
        - Green Power Common

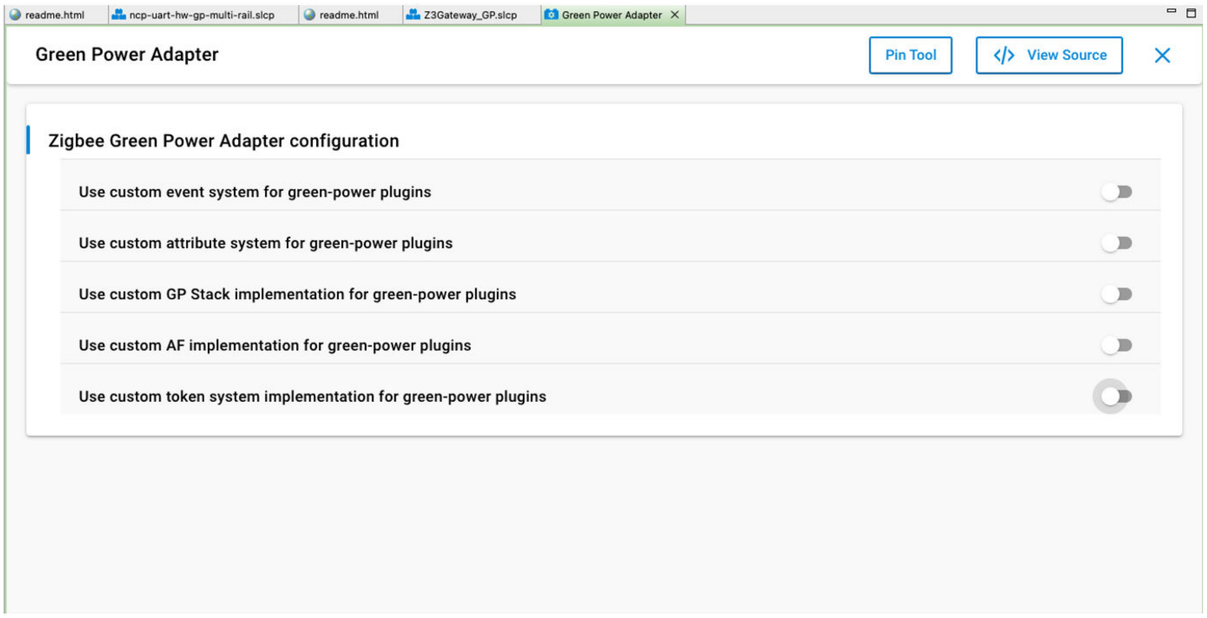
    b. Under Zigbee -> Green Power install the following components:
        - Green Power Client CLI: Implements the client-side CLI functionality of the Green Power Cluster
        - Green Power Server CLI: Implements the server-side CLI functionality of the Green Power Cluster
        - Green Power Translation Table CLI: Implements the Translation Table CLI functionality of the Green Power Cluster.
        - Application Framework Support Component: This Application Framework implementation is used for NCP applications.

    c. In the Configuration Tools tab open the Zigbee Cluster Configurator. Delete Endpoint 242 (Device – GP Combo Basic 0x0066).

    d. You can transfer your project to your platform and build it using a standard *make* command.

2. Using the **Zigbee - NCP UartHwGpMultiRail** sample application, create a new project. Open the Software Components in the Project Configurator:
   a. Install the following Components in Zigbee -> Green Power:
      - Application Framework Support Component.
      - Green Power Adapter: This component provides all the in/out interfaces for the green power cluster.
      - Green Power Client.
      - Green Power Combo Zap Config: If your NCP configuration is for a GPP you should use the Green Power Proxy Config Component.
      - Green Power Common.
      - Green Power Server.
      - Green Power Translation Table.

   b. Disable all Custom Options in the Green Power Adapter Component:



Green Power Adapter

Zigbee Green Power Adapter configuration

Use custom event system for green-power plugins

Use custom attribute system for green-power plugins

Use custom GP Stack implementation for green-power plugins

Use custom AF implementation for green-power plugins

Use custom token system implementation for green-power plugins

   c. Build and flash the project into your board.
3. Once you have built the Z3Gateway in step 1 in your host and flashed the Zigbee - NCP UartHwGpMultiRail application in step 2, you can connect the NCP to your host and run the Z3Gateway:

```
./Z3GatewayHost -p /dev/tty.usbmodem0004402507811
Reset info: 11 (SOFTWARE)
ezsp ver 0x0E stack type 0x02 stack ver.
[8.0.1 GA build 270]Ezsp Config: set address table size to 0x0002:Success: set
```

4. To pair the (Host + NCP) GPCombo with a GPD, on the GPCombo issue the CLI command

   plugin green-power-server commission 9 0 0 1: This will initiate commissioning in the Sink.

   Note: If a sink is setting its commissionin mode for groupcast, then it neds to ensure the binding table is not full and has enough available space for the commissioning endpoint to be added. This guarantees that packets will be properly forwarded.

   To send the GPD Command (Commission) from the GPD, issue the CLI command:

   node comm 255: This command will start commissioning and will eventually pair with the On-Off Cluster on Endpoint 1 on the Host.
5. Finally, you can send GPDF command 0x22 (see Table 4.1 Default Translation Table No Payload on page 11), which is the switch toggle CLI command for the GPD. This command will be received by the NCP and will be translated in the translation table to the on-off toggle command. The NCP will then ask the host to toggle the attribute of the On-Off Cluster in Endpoint 1.

6.3 Components

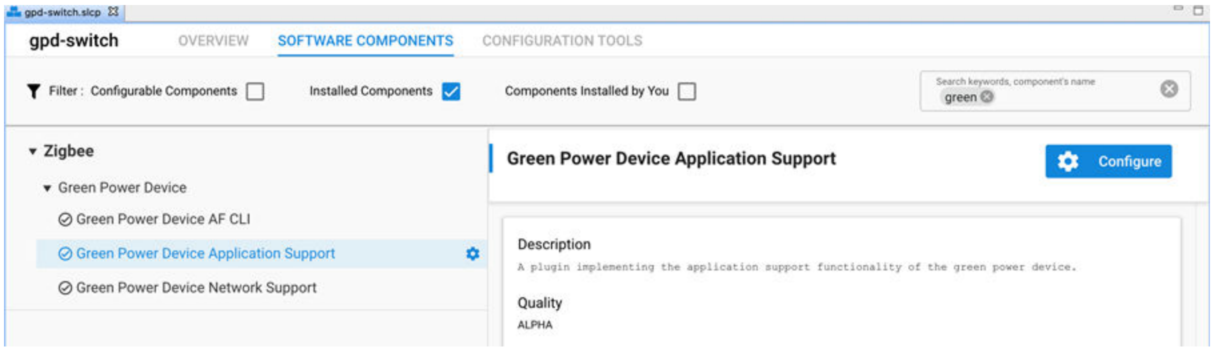
Components are made up of a collection of source files and properties. The component-based design enables customization by adding, configuring, and removing components. The application developer can use SSv5's Project Configurator and Component Editor to easily assemble the desired features by including those components that match the required functionality and by configuring the various properties associated with those components.

The configuration options and source code previously associated with plugins and the callbacks are all found as a component or part of a component.

The components commonly associated with Green Power functionality are the following:

Green Power Device:

- **Green Power Device AF CLI**: Implements the CLI for the Green Power Device application. Depends on the CLI service, GPD Application support, and Debug Print components.
- **Green Power Device Application Support**: Implements application layer support for the Green Power Device. Depends on the GPD Network Support component.
- **Green Power Device Network Support:** Implements network layer support for the Green Power Device.


gpd-switch.slcp
gpd-switch
OVERVIEW
SOFTWARE COMPONENTS
CONFIGURATION TOOLS
Filter : Configurable Components
Installed Components
Components Installed by You
Search keywords, component's name
green
Zigbee
Green Power Device
Green Power Device AF CLI
Green Power Device Application Support
Green Power Device Network Support
Configure

Green Power Device Application Support

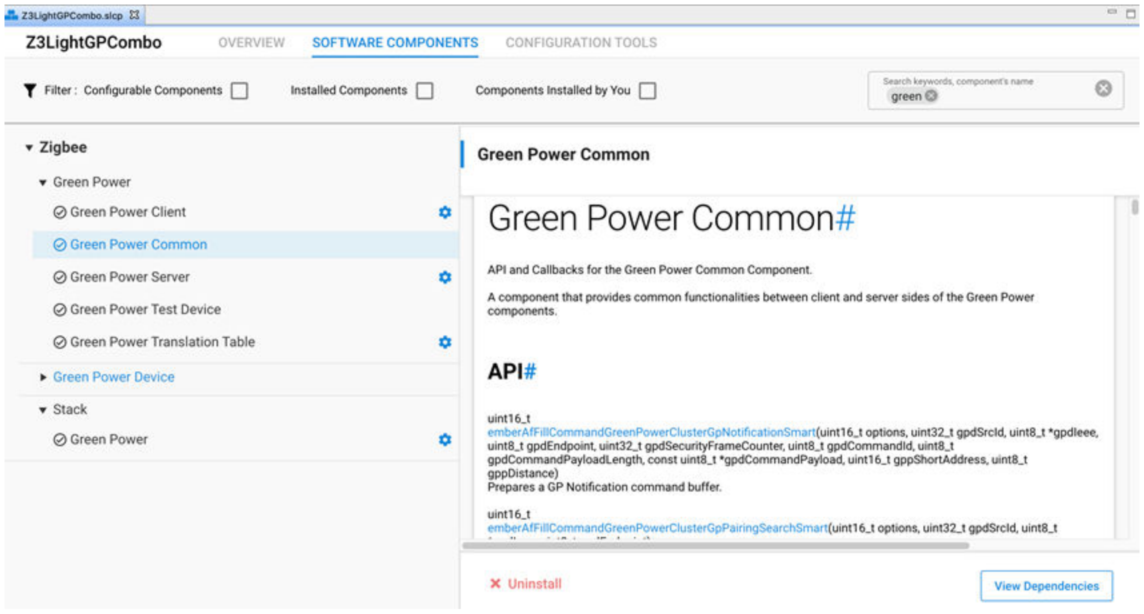
Description
A plugin implementing the application support functionality of the green power device.

Quality
ALPHA

Figure 6.2. Green Power Device Components

Green Power Sinks, Proxies and Combos

- **Application Framework Support Component**
  - Provides Application Framework implementation used on NCP applications.
- **Green Power Adapter**
  - Provides the green power cluster user with all the in/out function interfaces. The customer should be able to use their framework.
- **Green Power Client**
  - Implements the Green Power Client cluster from the Zigbee Cluster Library (ZCL).
  - Documentation can be found at: https://docs.silabs.com/zigbee/7.0/zigbee-af-api/green-power-client
- **Green Power Client CLI**
  - Implements the client-side CLI functionality of the Green Power Cluster.
- **Green Power Server**
  - Implements the Green Power Server cluster from the ZCL.
  - Documentation can be found at: https://docs.silabs.com/zigbee/7.0/zigbee-af-api/green-power-server
- **Green Power Server CLI**
  - Implements the server-side CLI functionality of the Green Power Cluster.
- **Green Power Common**
  - Contains code that is common between the server and client cluster implementations. Required by all Green Power applications within the Zigbee Framework.
  - Documentation can be found at: http://docs.silabs.com/zigbee/7.0/zigbee-af-api/green-power-common
- **Green Power Translation Table**
  - Implements the Green Power Translation Table.
- **Green Power Translation Table CLI**
  - Implements the translation table CLI functionality of the Green Power Cluster.
  - Documentation can be found at: https://docs.silabs.com/zigbee/7.0/zigbee-af-api/green-power-translation-table-cli
- **Stack - Green Power**
  - Implements the Green Power Stack within the Zigbee framework. Required by the Green Power Common component.



Green Power Common#

API and Callbacks for the Green Power Common Component.

A component that provides common functionalities between client and server sides of the Green Power components.

API#

```
uint16_t
emberAfFillCommandGreenPowerClusterGpNotificationSmart(uint16_t options, uint32_t gpdSrcId, uint8_t *gpdIeee,
uint8_t gpdEndpoint, uint32_t gpdSecurityFrameCounter, uint8_t gpdCommandId, uint8_t
gpdCommandPayloadLength, const uint8_t *gpdCommandPayload, uint16_t gppShortAddress, uint8_t
gppDistance)
```
Prepares a GP Notification command buffer.

```
uint16_t
emberAfFillCommandGreenPowerClusterGpPairingSearchSmart(uint16_t options, uint32_t gpdSrcId, uint8_t
```

Figure 6.3. Green Power Sink, Proxy and Combo Components

IoT Portfolio
www.silabs.com/products

Quality
www.silabs.com/quality

Support & Community
www.silabs.com/community

SILICON LABS

Silicon Laboratories Inc.
400 West Cesar Chavez
Austin, TX 78701
USA

www.silabs.com