# Z-Stack
# ZigBee Cluster Library
# Application Programming Interface

Document Number: SWRA197

**Texas Instruments, Inc.**
San Diego, California USA

| Version | Description | Date |
|---|---|---|
| 1.0 | Initial release | 12/07/2006 |
| 1.1 | Added Compile Options | 03/20/2007 |
| 1.2 | Updated as per latest specs (Foundation 06027r06 and General 053936r05) | 05/02/2007 |
| 1.3 | Updated as per latest spec (075123r01ZB) | 11/28/2007 |
| 1.4 | Removed Logical Cluster IDs | 03/31/2009 |
| 1.5 | Added Protocol Interfaces functional domain | 10/26/2009 |
| 1.6 | Added 11073 Protocol Tunnel Cluster | 12/18/2009 |
| 1.7 | Added interface to retrieve raw AF message | 04/05/2010 |
| 1.8 | Updated 11073 Protocol Tunnel Cluster | 09/29/2010 |
| 1.9 | Added Closures functional domain,<br>Added zclGeneral_ReadSceneCountCB(),<br>Removed Smart Energy specific options from Compile Options | 07/18/2011 |
| 1.10 | Added Touchlink Commissioning<br>Added Green Power | 11/20/2016 |
| 1.11 | Addition of ZCL_DOORLOCK_EXT for optional commands of doorlock | 10/5/2017 |

## TABLE OF CONTENTS



 

 

 

 

## LIST OF FIGURES



 

# 1. Introduction

## 1.1 Purpose

The purpose of this document is to define the ZigBee Cluster Library (ZCL) API. This API allows the higher layers (Profile and Application) to access the ZCL functionality. The ZCL is divided into the Foundation layer and several functional domains, each domain addressing clusters relating to specific functionality. The functional domains are:

- General
- Closures
- Heating, Ventilation and Air Conditioning (HVAC)
- Lighting
- Measurements and Sensing
- Security and Safety
- Smart Energy
- Protocol Interfaces
- Touchlink Commissioning
- Green Power

This document covers only the Foundation layer and the General, Protocol Interfaces, Touchlink Commissioning and Green Power functional domains.

## 1.2 Scope

This document enumerates all the function calls provided by the Foundation layer, Touchlink Commissioning and Green Power General functional domain. It also enumerates the callback functions that need to be provided by the higher layers.

## 1.3 Acronyms

| | |
|---|---|
| AF | Application Framework |
| APDU | Application Protocol Data Unit |
| API | Application Programming Interface |
| APS | Application Support Sub-Layer |
| BACnet | Building Automation and Control Network |
| Client | A cluster interface which is listed in the output cluster list of the simple descriptor on an endpoint. Typically this interface sends commands that manipulate the attributes on the corresponding server cluster |
| Cluster | A related collection of attributes and commands, which together define a communications interface between two devices. The devices implement server and client sides of the interface respectively |
| HAL | Hardware Abstraction Layer |
| MAC | Media Access Control |
| NPDU | Network Protocol Data Unit |
| NWK | Network Layer |
| PAN | Personal Area Network |
| RSSI | Receiver Signal Strength Indication |
| Server | A cluster interface which is listed in the input cluster list of the simple descriptor on an endpoint. Typically this interface supports all or most of the attributes of the cluster |
| ZCL | ZigBee Cluster Library |

 

### 1.4 Applicable Documents

1. ZigBee document 053520r16, ZigBee Profile: Home Automation, ZigBee Application Framework Working Group
2. ZigBee document 06027r04, ZigBee Cluster Library, Foundation, ZigBee Application Framework Working Group
3. ZigBee document 053936r04a, ZigBee Cluster Library, Functional Domain: General, ZigBee Application Framework Working Group
4. ZigBee document 075123r06ZB, ZigBee Cluster Library Specification
5. Zigbee document 14-0563-16 Green Power Basic specification
6. Texas Instruments document SWRA216, Z-Stack Smart Energy Developer's Guide
7. Texas Instruments document SWRA195 Z-Stack Application Programming Interface

## 2. API Overview

### 2.1 Overview

The ZCL acts as a repository for cluster functionality that is developed by ZigBee. The Foundation layer and General functional domain cluster functionality is covered in this document.
The Foundation layer provides APIs to the higher layers to:

1. Generate Request and Response commands
2. Register Application's attribute list
3. Register Application's attribute data validation callback function
4. Register Cluster Library Handler callback functions
5. Register the Application task to receive the unprocessed Foundation command/response messages

The General and Protocol Interfaces functional domains provide APIs to the high layers to:

1. Generate Request and Response commands
2. Register Application's Command callback functions

### 2.2 Client/Server Model

The ZCL employs a Client/Server model. A *cluster* is a related collection of commands and attributes, which together define an interface to specific functionality. Typically, the entity that stores the attributes of a cluster is referred to as the *Server*, and an entity that affects or manipulates those attributes is referred as the *Client*. However, if required, attributes may also be present on the Client of a cluster.

As an example, the Read and Write attribute commands, which allow devices to manipulate attributes, are sent from the Client device and received by the Server device. Any response to those commands (i.e., the Read and Write attribute response commands) are sent from the Server device and received by the Client device. Conversely, the Report attribute command, which facilitates dynamic attribute reporting, is sent from the Server device to the Client device that has been bound to the Server device.

 

### 2.3 Stack Diagram

Figure 1 illustrates the components within the ZCL and its interfaces with other layers.

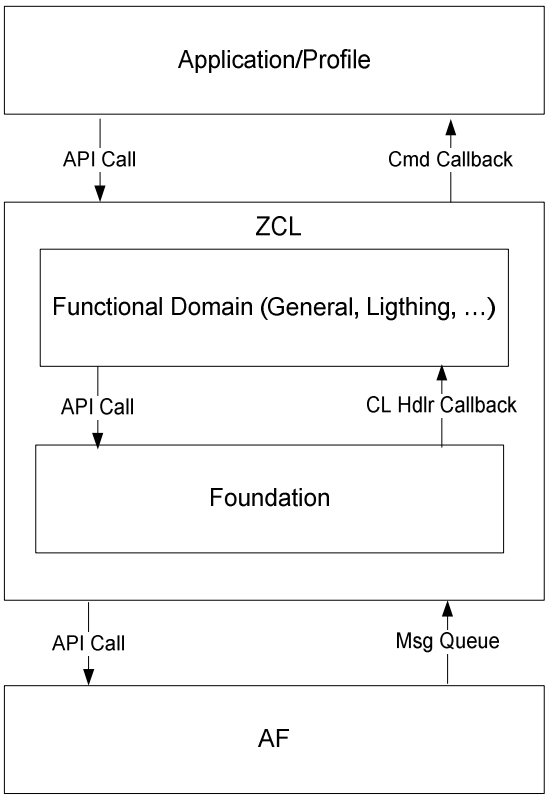


**Figure 1: Stack Diagram**

The ZCL command messages received by the AF are put into the ZCL task's queue. The ZCL task parses and processes the *profile* commands, and hands the *cluster-specific* commands off to the corresponding cluster through Cluster Library Handler callback function. The cluster processes the command and notifies, if required, the application/profile through Command callback function.

 

### 2.4 Application/Profile Registration

The Foundation provides `zcl_registerAttrList()`, `zcl_registerValidateAttrData()` and `zcl_registerForMsg()` APIs to the Application/Profile to register Application's attribute list, attribute data validation callback and Task ID respectively. It also provides `zcl_registerPlugin()` API to the functional domains to register their Cluster Library Handler callback function. The prototype of the callback function is defined in section 3.20.

The attribute list input parameter to `zcl_registerAttrList()` contains an entry of the following information for each supported attribute:

```
// Attribute record
typedef struct
{
  uint16  attrId;         // Attribute ID
  uint8   dataType;       // Data Type - defined in AF.h
  uint8   accessControl;  // Read/write - bit field
  void    *dataPtr;       // Pointer to data field
} zclAttribute_t;
typedef struct
{
  uint16          clusterID;  // Cluster ID
  zclAttribute_t  attr;       // Attribute record
} zclAttrRec_t;
```

The parameter list of `zcl_registerValidateAttrData()` contains pointers to the application's attribute record (`zclAttrRec_t`) and the new data to be written (`zclWriteRec_t`):

```
// Write Attribute record
typedef struct
{
  uint16 attrID;          // attribute ID
  uint8  dataType;        // attribute data type
  uint8  *attrData;       // this structure is allocated, so the data is HERE
                          //  - the size depends on the attribute data type
} zclWriteRec_t;
```

The General functional domain provides `zclGeneral_RegisterCmdCallbacks()` API to register Application's Command callback functions. The command callback input parameter to this API is of the following type:

```
// Register Callbacks table entry - enter function pointers for callbacks that
// the application would like to receive

typedef struct
{
  zclGCB_BasicReset_t                     pfnBasicReset;
  zclGCB_IdentifyTriggerEffect_t          pfnIdentifyTriggerEffect;
  zclGCB_OnOff_t                          pfnOnOff;
  zclGCB_OnOff_OffWithEffect_t            pfnOnOff_OffWithEffect;
  zclGCB_OnOff_OnWithRecallGlobalScene_t  pfnOnOff_OnWithRecallGlobalScene;
  zclGCB_OnOff_OnWithTimedOff_t           pfnOnOff_OnWithTimedOff;

  zclGCB_LevelControlMoveToLevel_t        pfnLevelControlMoveToLevel;
  zclGCB_LevelControlMove_t               pfnLevelControlMove;
  zclGCB_LevelControlStep_t               pfnLevelControlStep;
  zclGCB_LevelControlStop_t               pfnLevelControlStop;
  zclGCB_GroupRsp_t                       pfnGroupRsp;
  zclGCB_SceneStoreReq_t                  pfnSceneStoreReq;
```

 

```
  zclGCB_SceneRecallReq_t               pfnSceneRecallReq;
  zclGCB_SceneRsp_t                     pfnSceneRsp;
  zclGCB_Alarm_t                        pfnAlarm;
  zclGCB_GetEventLog_t                  pfnGetEventLog;
  zclGCB_PublishEventLog_t              pfnPublishEventLog;
  zclGCB_Location_t                     pfnLocation;
  zclGCB_LocationRsp_t                  pfnLocationRsp;
} zclGeneral_AppCallbacks_t;
```

The prototype of each command callback function is defined in section 4.

```
Note 1: Some of this callbacks are defined by compilation flags such as
ZCL_GROUPS, please refer to the structure implementation in zcl_general.h
Note 2: The identify callbacks are now handled by BDB, please refer to [7] to
see how to manage identify interface.
```

The Closures functional domain provides `zclClosures_RegisterDoorLockCmdCallbacks()` and `zclClosures_RegisterWindowCoveringCmdCallbacks()` API to register Application's Command callback functions. The command callback input parameter to this API is of the following type:

// Register Callbacks table entry - enter function pointers for callbacks that
// the application would like to receive

```
typedef struct
{
  zclClosures_DoorLock_t                        pfnDoorLock;
  zclClosures_DoorLockRsp_t                     pfnDoorLockRsp;
  zclClosures_DoorLockUnlockWithTimeout_t       pfnDoorLockUnlockWithTimeout;
  zclClosures_DoorLockGetLogRecord_t            pfnDoorLockGetLogRecord;
  zclClosures_DoorLockSetPINCode_t              pfnDoorLockSetPINCode;
  zclClosures_DoorLockGetPINCode_t              pfnDoorLockGetPINCode;
  zclClosures_DoorLockClearPINCode_t            pfnDoorLockClearPINCode;
  zclClosures_DoorLockClearAllPINCodes_t        pfnDoorLockClearAllPINCodes;
  zclClosures_DoorLockSetUserStatus_t           pfnDoorLockSetUserStatus;
  zclClosures_DoorLockGetUserStatus_t           pfnDoorLockGetUserStatus;
  zclClosures_DoorLockSetWeekDaySchedule_t      pfnDoorLockSetWeekDaySchedule;
  zclClosures_DoorLockGetWeekDaySchedule_t      pfnDoorLockGetWeekDaySchedule;
  zclClosures_DoorLockClearWeekDaySchedule_t    pfnDoorLockClearWeekDaySchedule;
  zclClosures_DoorLockSetYearDaySchedule_t      pfnDoorLockSetYearDaySchedule;
  zclClosures_DoorLockGetYearDaySchedule_t      pfnDoorLockGetYearDaySchedule;
  zclClosures_DoorLockClearYearDaySchedule_t    pfnDoorLockClearYearDaySchedule;
  zclClosures_DoorLockSetHolidaySchedule_t      pfnDoorLockSetHolidaySchedule;
  zclClosures_DoorLockGetHolidaySchedule_t      pfnDoorLockGetHolidaySchedule;
  zclClosures_DoorLockClearHolidaySchedule_t    pfnDoorLockClearHolidaySchedule;
  zclClosures_DoorLockSetUserType_t             pfnDoorLockSetUserType;
  zclClosures_DoorLockGetUserType_t             pfnDoorLockGetUserType;
  zclClosures_DoorLockSetRFIDCode_t             pfnDoorLockSetRFIDCode;
  zclClosures_DoorLockGetRFIDCode_t             pfnDoorLockGetRFIDCode;
  zclClosures_DoorLockClearRFIDCode_t           pfnDoorLockClearRFIDCode;
  zclClosures_DoorLockClearAllRFIDCodes_t       pfnDoorLockClearAllRFIDCodes;
  zclClosures_DoorLockUnlockWithTimeoutRsp_t    pfnDoorLockUnlockWithTimeoutRsp;
  zclClosures_DoorLockGetLogRecordRsp_t         pfnDoorLockGetLogRecordRsp;
  zclClosures_DoorLockSetPINCodeRsp_t           pfnDoorLockSetPINCodeRsp;
  zclClosures_DoorLockGetPINCodeRsp_t           pfnDoorLockGetPINCodeRsp;
  zclClosures_DoorLockClearPINCodeRsp_t         pfnDoorLockClearPINCodeRsp;
  zclClosures_DoorLockClearAllPINCodesRsp_t     pfnDoorLockClearAllPINCodesRsp;
  zclClosures_DoorLockSetUserStatusRsp_t        pfnDoorLockSetUserStatusRsp;
  zclClosures_DoorLockGetUserStatusRsp_t        pfnDoorLockGetUserStatusRsp;
  zclClosures_DoorLockSetWeekDayScheduleRsp_t   pfnDoorLockSetWeekDayScheduleRsp;
  zclClosures_DoorLockGetWeekDayScheduleRsp_t   pfnDoorLockGetWeekDayScheduleRsp;
  zclClosures_DoorLockClearWeekDayScheduleRsp_t pfnDoorLockClearWeekDayScheduleRsp;
  zclClosures_DoorLockSetYearDayScheduleRsp_t   pfnDoorLockSetYearDayScheduleRsp;
  zclClosures_DoorLockGetYearDayScheduleRsp_t   pfnDoorLockGetYearDayScheduleRsp;
  zclClosures_DoorLockClearYearDayScheduleRsp_t pfnDoorLockClearYearDayScheduleRsp;
  zclClosures_DoorLockSetHolidayScheduleRsp_t   pfnDoorLockSetHolidayScheduleRsp;
  zclClosures_DoorLockGetHolidayScheduleRsp_t   pfnDoorLockGetHolidayScheduleRsp;
  zclClosures_DoorLockClearHolidayScheduleRsp_t pfnDoorLockClearHolidayScheduleRsp;
```

 

```
  zclClosures_DoorLockSetUserTypeRsp_t                pfnDoorLockSetUserTypeRsp;
  zclClosures_DoorLockGetUserTypeRsp_t                pfnDoorLockGetUserTypeRsp;
  zclClosures_DoorLockSetRFIDCodeRsp_t                pfnDoorLockSetRFIDCodeRsp;
  zclClosures_DoorLockGetRFIDCodeRsp_t                pfnDoorLockGetRFIDCodeRsp;
  zclClosures_DoorLockClearRFIDCodeRsp_t              pfnDoorLockClearRFIDCodeRsp;
  zclClosures_DoorLockClearAllRFIDCodesRsp_t          pfnDoorLockClearAllRFIDCodesRsp;
  zclClosures_DoorLockOperationEventNotification_t    pfnDoorLockOperationEventNotification;
  zclClosures_DoorLockProgrammingEventNotification_t  pfnDoorLockProgrammingEventNotification;
} zclClosures_DoorLockAppCallbacks_t;

typedef struct
{
  zclClosures_WindowCoveringSimple_t             pfnWindowCoveringUpOpen;
  zclClosures_WindowCoveringSimple_t             pfnWindowCoveringDownClose;
  zclClosures_WindowCoveringSimple_t             pfnWindowCoveringStop;
  zclClosures_WindowCoveringGotoValue_t          pfnWindowCoveringGotoLiftValue;
  zclClosures_WindowCoveringGotoPercentage_t     pfnWindowCoveringGotoLiftPercentage;
  zclClosures_WindowCoveringGotoValue_t          pfnWindowCoveringGotoTiltValue;
  zclClosures_WindowCoveringGotoPercentage_t     pfnWindowCoveringGotoTiltPercentage;
} zclClosures_WindowCoveringAppCallbacks_t;
```

The prototype of each command callback function is defined in section 5.

The Touchlink Commissioning provides `zclPI_RegisterCmdCallbacks()` API to register Application's Command callback functions. The command callback input parameter to this API is of the following type:

// Register Callbacks table entry - enter function pointers for callbacks that
// the application would like to receive

```
typedef struct
{
  zclPICB_MatchProtocolAddr_t      pfnPI_MatchProtocolAddr;
  zclPICB_MatchProtocolAddrRsp_t   pfnPI_MatchProtocolAddrRsp;
  zclPICB_AdvertiseProtocolAddr_t  pfnPI_AdvertiseProtocolAddr;
  zclPICB_BACnetTransferNPDU_t     pfnPI_BACnetTransferNPDU;
  zclPICB_11073TransferAPDU_t      pfnPI_11073TransferAPDU;
  zclPICB_11073ConnectReq_t        pfnPI_11073ConnectReq;
  zclPICB_11073DisconnectReq_t     pfnPI_11073DisconnectReq;
  zclPICB_11073ConnectStatusNoti_t pfnPI_11073ConnectStatusNoti;
} zclPI_AppCallbacks_t;
```

The prototype of each command callback function is defined in section 6.

The Protocol Interfaces functional domain provides `bdbTL_RegisterCmdCallbacks()` `bdbTL_RegisterInterPANCmdCallbacks()` API to register Application's Command callback functions. The command callback input parameter to this API is of the following type:

// Register Callbacks table entry - enter function pointers for callbacks that
// the application would like to receive

```
typedef struct
{
  bdbTL_GetGrpIDsReqCB_t   pfnGetGrpIDsReq;
  bdbTL_GetEPListReqCB_t   pfnGetEPListReq;
  bdbTL_EndpointInfoCB_t   pfnEndpointInfo;
  bdbTL_GetGrpIDsRspCB_t   pfnGetGrpIDsRsp;
  bdbTL_GetEPListRspCB_t   pfnGetEPListRsp;
} bdbTL_AppCallbacks_t;

typedef struct
{
  bdbTL_ScanReqCB_t          pfnScanReq;
  bdbTL_DeviceInfoReqCB_t    pfnDeviceInfoReq;
  bdbTL_IdentifyReqCB_t      pfnIdentifyReq;
  bdbTL_ResetToFNReqCB_t     pfnResetToFNReq;
  bdbTL_NwkStartReqCB_t      pfnNwkStartReq;
```

 

```
  bdbTL_NwkJoinRtrReqCB_t    pfnNwkJoinRtrReq;
  bdbTL_NwkJoinEDReqCB_t     pfnNwkJoinEDReq;
  bdbTL_NwkUpdateReqCB_t     pfnNwkUpdateReq;
  bdbTL_ScanRspCB_t          pfnScanRsp;
  bdbTL_DeviceInfoRspCB_t    pfnDeviceInfoRsp;
  bdbTL_NwkStartRspCB_t      pfnNwkStartRsp;
  bdbTL_NwkJoinRtrRspCB_t    pfnNwkJoinRtrRsp;
  bdbTL_NwkJoinEDRspCB_t     pfnNwkJoinEDRsp;
} bdbTL_InterPANCallbacks_t;
```

The prototype of each command callback function is defined in section 7.

### 2.5 Application Creation

This section outlines the steps to be taken when creating a new ZCL application. At least four modules should be created for the new application:

- zcl_<appname>.h which should contain the definitions needed for the application
- zcl_<appname>_data.c which should contain the data definitions and declarations needed for the application
- zcl_<appname>.c which should contain all the functions and callback functions needed for the application
- OSAL_<AppName>.c where all the tasks needed for the application should be added to the task list

Each module is explained in detail in the following subsections.

#### 2.5.1 zcl_<appname>.h

This header file should contain all the definitions needed for the new application. The application's *endpoint* should be defined in this module.

#### 2.5.2 zcl_<appname>_data.c

This module should contain the declaration of:

1. All the *cluster attributes* that are supported by the application
2. The *attribute table* containing one entry of `zclAttrRec_t` type for each supported attribute
3. The *input* and *output cluster ID tables*, where these tables are filled with the application-specific input and output cluster IDs respectively. These tables are used with the simple descriptor table
4. The application's *simple descriptor table* of `SimpleDescriptionFormat_t` type defined in AF.h header file

#### 2.5.3 zcl_<appname>.c

This module should contain the following items:

1. The declaration of the application's *endpoint table* of `endPointDesc_t` type defined in AF.h header file
2. Create all the *command callback functions* to handle any incoming command from the ZCL clusters. These callback functions are used with the command callback tables
3. The declaration of the application's *command callback tables* for the ZCL functional domains. The type of this table for the General functional domain is `zclGeneral_AppCallbacks_t`, which is defined in zcl_general.h header file
4. Create `void zcl<AppName>_Init( byte task_id )` function for the application task. This function's responsibility is listed below
5. Create `uint16 zcl<AppName>_event_loop( uint8 task_id, uint16 events )` function to receive and process messages and key events put on the application task's queue

The application's initialization function `zcl<AppName>_Init()` should register:

1. The *command callback tables* with the corresponding functional domains. The function `zclGeneral_RegisterCmdCallbacks()` defined in zcl_general.c module should be used to register the General cluster command callbacks
2. The application's *attribute list* with ZCL Foundation using `zcl_registerAttrList()` API which is defined in zcl.c module
3. The application's *endpoint* with the AF layer using `afRegister()` API defined in AF.c module
4. The application's *task* with the hardware to process all "press-key" events using `RegisterForKeys()` API defined in OnBoard.c module (if the application handles any key event)

 

5. The application's *simple descriptor* with the HA profile using `bdb_RegisterSimpleDescriptor()` API defined in bdb.c module

### 2.5.4 OSAL_<AppName>.c

This module should contain `void osalInitTasks ( void )` function, where all the tasks needed for the application and the application task itself are added to the task list. The task addition is done using passing each application an Application ID that increases over each initialized task (refer to any sample application). Here's the list of tasks and their addition order needed for a simple ZCL application:

1. MAC
2. Network
3. Green Power (in the case of Router and Coordinator devices)
4. HAL
5. MT (If needed by the application)
6. APS
7. APS Fragmentation
8. ZD Application
9. ZD Network Manager (If needed by the application)
10. ZCL
11. BDB
12. ZCL Application

*Note:* Changing the order of this tasks will affect the of the stack processing producing unpredictable behavior. Only the tasks with comments are optional under the comments considerations.

 

# 3. Foundation Layer

## 3.1 Introduction

The Foundation layer provides general commands that are used for manipulating attributes and other general tasks that are not specific to an individual cluster. These commands are:

- Read attributes
- Read attributes response
- Write attributes
- Write attributes undivided
- Write attributes response
- Write attributes no response
- Configure reporting
- Configure reporting response
- Read reporting configuration
- Read reporting configuration response
- Report attributes
- Default response
- Discover attributes
- Discover attributes response

## 3.2 Send Command

### 3.2.1 Description

This function is used to send Profile and Cluster Specific Command messages.

### 3.2.2 Prototype

```
ZStatus_t zcl_SendCommand( uint8 srcEP, afAddrType_t *destAddr,
                           uint16 clusterID, uint8 cmd, uint8 specific,
                           uint8 direction, uint8 disableDefaultRsp,
                           uint16 manuCode, uint8 seqNum, uint8 cmdFormatLen,
                           uint8 *cmdFormat );
```

### 3.2.3 Parameter Details

`srcEP` – The source endpoint.
`destAddr` – The destination address.
`clusterID` – The identifier of the cluster.
`cmd` - The command ID.
`specific` - Whether the command is Cluster Specific.
`direction` - The client/server direction of the command.
`disableDefaultRsp` - Disable Default Response command.
`manuCode` - The manufacturer code for proprietary extensions to a profile.
`seqNum` – The identification number for the transaction.
`cmdFormatLen` - The length of the command to be sent.
`cmdFormat` – The command to be sent.

### 3.2.4 Return

`ZStatus_t` – enum found in ZComDef.h.

 

## 3.3 Send Read

### 3.3.1 Description
This function is used to send a Read Attributes command.

### 3.3.2 Prototype
```
ZStatus_t zcl_SendRead( uint8 srcEP, afAddrType_t *destAddr,
uint16 clusterID, zclReadCmd_t *readCmd,
uint8 direction, uint8 disableDefaultRsp,
uint8 seqNum );
```

### 3.3.3 Parameter Details

`srcEP` – The source endpoint.
`destAddr` – The destination address.
`clusterID` – The identifier of the cluster.
`readCmd` - The Read command to be sent.
`direction` - The client/server direction of the command.
`disableDefaultRsp` - Disable Default Response command.
`seqNum` - The identification number for the transaction.

### 3.3.4 Return
`ZStatus_t` – enum found in ZComDef.h.

## 3.4 Send Read Response

### 3.4.1 Description
This function is used to send a Read Attributes Response command.

### 3.4.2 Prototype
```
ZStatus_t zcl_SendReadRsp( uint8 srcEP, afAddrType_t *destAddr,
                           uint16 clusterID, zclReadRspCmd_t *readRspCmd,
                           uint8 direction, uint8 disableDefaultRsp,
                           uint8 seqNum );
```

### 3.4.3 Parameter Details
`srcEP` – The source endpoint.
`destAddr` – The destination address.
`clusterID` – The identifier of the cluster.
`readRspCmd` - The Read Response command to be sent.
`direction` - The client/server direction of the command.
`disableDefaultRsp` - Disable Default Response command.
`seqNum` - The identification number for the transaction.

### 3.4.4 Return
`ZStatus_t` – enum found in ZComDef.h.

 

### 3.5 Send Write

#### 3.5.1 Description
This function is used to send a Write Attributes command.

#### 3.5.2 Prototype
```
ZStatus_t zcl_SendWrite( uint8 srcEP, afAddrType_t *destAddr,
                         uint16 clusterID, zclWriteCmd_t *writeCmd,
                         uint8 direction, uint8 disableDefaultRsp,
                         uint8 seqNum );
```

#### 3.5.3 Parameter Details
`srcEP` – The source endpoint.
`destAddr` – The destination address.
`clusterID` – The identifier of the cluster.
`writeCmd` - The Write command to be sent.
`direction` - The client/server direction of the command.
`disableDefaultRsp` - Disable Default Response command.
`seqNum` - The identification number for the transaction.

#### 3.5.4 Return
`ZStatus_t` – enum found in ZComDef.h.

### 3.6 Send Write Undivided

#### 3.6.1 Description
This function is used to send a Write Attributes Undivided command.

#### 3.6.2 Prototype
```
ZStatus_t zcl_SendWriteUndivided( uint8 srcEP, afAddrType_t *destAddr,
                                  uint16 clusterID, zclWriteCmd_t *writeCmd,
                                  uint8 direction, uint8 disableDefaultRsp,
                                  uint8 seqNum );
```

#### 3.6.3 Parameter Details
`srcEP` – The source endpoint.
`destAddr` – The destination address.
`clusterID` – The identifier of the cluster.
`writeCmd` - The Write Undivided command to be sent.
`direction` - The client/server direction of the command.
`disableDefaultRsp` - Disable Default Response command.
`seqNum` - The identification number for the transaction.

#### 3.6.4 Return
`ZStatus_t` – enum found in ZComDef.h.

 

## 3.7 Send Write Response

### 3.7.1 Description
This function is used to send a Write Attributes Response command.

### 3.7.2 Prototype
```
ZStatus_t zcl_SendWriteRsp( uint8 srcEP, afAddrType_t *destAddr,
                            uint16 clusterID, zclWriteRspCmd_t *writeRspCmd,
                            uint8 direction, uint8 disableDefaultRsp,
                            uint8 seqNum );
```

### 3.7.3 Parameter Details
`srcEP` – The source endpoint.
`destAddr` – The destination address.
`clusterID` – The identifier of the cluster.
`writeRspCmd` - The Write Response command to be sent.
`direction` - The client/server direction of the command.
`disableDefaultRsp` - Disable Default Response command.
`seqNum` - The identification number for the transaction.

### 3.7.4 Return
`ZStatus_t` – enum found in ZComDef.h.

## 3.8 Send Write No Response

### 3.8.1 Description
This function is used to send a Write Attributes No Response command.

### 3.8.2 Prototype
```
ZStatus_t zcl_SendWriteNoRsp( uint8 srcEP, afAddrType_t *destAddr,
                              uint16 clusterID, zclWriteCmd_t *writeCmd,
                              uint8 direction, uint8 disableDefaultRsp,
                              uint8 seqNum );
```

### 3.8.3 Parameter Details
`srcEP` – The source endpoint.
`destAddr` – The destination address.
`clusterID` – The identifier of the cluster.
`writeCmd` - The Write No Response command to be sent.
`direction` - The client/server direction of the command.
`disableDefaultRsp` - Disable Default Response command.
`seqNum` - The identification number for the transaction.

### 3.8.4 Return
`ZStatus_t` – enum found in ZComDef.h.

 

## 3.9 Send Configure Reporting

### 3.9.1 Description
This function is used to send a Configure Reporting command.

### 3.9.2 Prototype
```
ZStatus_t zcl_SendConfigReportCmd( uint8 srcEP, afAddrType_t *destAddr,
                    uint16 clusterID, zclCfgReportCmd_t *cfgReportCmd,
                    uint8 direction, uint8 disableDefaultRsp, uint8 seqNum );
```

### 3.9.3 Parameter Details
`srcEP` – The source endpoint.
`destAddr` – The destination address.
`clusterID` – The identifier of the cluster.
`cfgReportCmd` - The Configure Reporting command to be sent.
`direction` - The client/server direction of the command.
`disableDefaultRsp` - Disable Default Response command.
`seqNum` - The identification number for the transaction.

### 3.9.4 Return
`ZStatus_t` – enum found in ZComDef.h.

## 3.10 Send Configure Reporting Response

### 3.10.1 Description
This function is used to send a Configure Reporting Response command.

### 3.10.2 Prototype

```
ZStatus_t zcl_SendConfigReportRspCmd( uint8 srcEP, afAddrType_t *destAddr,
                 uint16 clusterID, zclCfgReportRspCmd_t *cfgReportRspCmd,
                 uint8 direction, uint8 disableDefaultRsp, uint8 seqNum );
```

### 3.10.3 Parameter Details
`srcEP` – The source endpoint.
`destAddr` – The destination address.
`clusterID` – The identifier of the cluster.
`cfgReportRspCmd` - The Configure Reporting Response command to be sent.
`direction` - The client/server direction of the command.
`disableDefaultRsp` - Disable Default Response command.
`seqNum` - The identification number for the transaction.

### 3.10.4 Return
`ZStatus_t` – enum found in ZComDef.h.

 

## 3.11 Send Read Reporting Configuration

### 3.11.1 Description
This function is used to send a Read Reporting Configuration command.

### 3.11.2 Prototype

```
ZStatus_t zcl_SendReadReportCfgCmd( uint8 srcEP, afAddrType_t *destAddr,
               uint16 clusterID, zclReadReportCfgCmd_t *readReportCfgCmd,
               uint8 direction, uint8 disableDefaultRsp, uint8 seqNum );
```

### 3.11.3 Parameter Details
`srcEP` – The source endpoint.
`destAddr` – The destination address.
`clusterID` – The identifier of the cluster.
`readReportCfgCmd` - The Read Reporting Configuration command to be sent.
`direction` - The client/server direction of the command.
`disableDefaultRsp` - Disable Default Response command.
`seqNum` - The identification number for the transaction.

### 3.11.4 Return
`ZStatus_t` – enum found in ZComDef.h.

## 3.12 Send Read Reporting Configuration Response

### 3.12.1 Description
This function is used to send a Read Reporting Configuration Response command.

### 3.12.2 Prototype

```
ZStatus_t zcl_SendReadReportCfgRspCmd( uint8 srcEP, afAddrType_t *destAddr,
         uint16 clusterID, zclReadReportCfgRspCmd_t *readReportCfgRspCmd,
         uint8 direction, uint8 disableDefaultRsp, uint8 seqNum );
```

### 3.12.3 Parameter Details
`srcEP` – The source endpoint.
`destAddr` – The destination address.
`clusterID` – The identifier of the cluster.
`readReportCfgRspCmd` - The Read Reporting Configuration Response command to be sent.
`direction` - The client/server direction of the command.
`disableDefaultRsp` - Disable Default Response command.
`seqNum` - The identification number for the transaction.

### 3.12.4 Return
`ZStatus_t` – enum found in ZComDef.h.

 

## 3.13 Send Report

### 3.13.1 Description
This function is used to send a Report Attributes command.

### 3.13.2 Prototype

```
ZStatus_t zcl_SendReportCmd( uint8 srcEP, afAddrType_t *destAddr,
                             uint16 clusterID, zclReportCmd_t *reportCmd,
                             uint8 direction, uint8 disableDefaultRsp,
                             uint8 seqNum );
```

### 3.13.3 Parameter Details
`srcEP` – The source endpoint.
`destAddr` – The destination address.
`clusterID` – The identifier of the cluster.
`reportCmd` - The Report command to be sent.
`direction` - The client/server direction of the command.
`disableDefaultRsp` - Disable Default Response command.
`seqNum` - The identification number for the transaction.

### 3.13.4 Return
`ZStatus_t` – enum found in ZComDef.h.

## 3.14 Send Default Response

### 3.14.1 Description
This function is used to send a Default Response command.

### 3.14.2 Prototype

```
ZStatus_t zcl_SendDefaultRspCmd( uint8 srcEP, afAddrType_t *destAddr,
                    uint16 clusterID, zclDefaultRspCmd_t *defaultRspCmd,
                    uint8 direction, uint8 disableDefaultRsp, uint8 seqNum );
```

### 3.14.3 Parameter Details
`srcEP` – The source endpoint.
`destAddr` – The destination address.
`clusterID` – The identifier of the cluster.
`defaultRspCmd` - The Default Response command to be sent.
`direction` - The client/server direction of the command.
`disableDefaultRsp` - Disable Default Response command.
`seqNum` - The identification number for the transaction.

### 3.14.4 Return
`ZStatus_t` – enum found in ZComDef.h.

 

### 3.15 Send Discover

#### 3.15.1 Description
This function is used to send a Discover Attributes command.

#### 3.15.2 Prototype

```
ZStatus_t zcl_SendDiscoverCmd( uint8 srcEP, afAddrType_t *destAddr,
                          uint16 clusterID, zclDiscoverCmd_t *discoverCmd,
                          uint8 direction, uint8 disableDefaultRsp,
                          uint8 seqNum );
```

#### 3.15.3 Parameter Details
`srcEP` – The source endpoint.
`destAddr` – The destination address.
`clusterID` – The identifier of the cluster.
`discoverCmd` - The Discover command to be sent.
`direction` - The client/server direction of the command.
`disableDefaultRsp` - Disable Default Response command.
`seqNum` - The identification number for the transaction.

#### 3.15.4 Return
`ZStatus_t` – enum found in ZComDef.h.

### 3.16 Send Discover Response

#### 3.16.1 Description
This function is used to send a Discover Attributes Response command.

#### 3.16.2 Prototype

```
ZStatus_t zcl_SendDiscoverRspCmd( uint8 srcEP, afAddrType_t *destAddr,
                    uint16 clusterID, zclDiscoverRspCmd_t *discoverRspCmd,
                    uint8 direction, uint8 disableDefaultRsp, uint8 seqNum );
```

#### 3.16.3 Parameter Details
`srcEP` – The source endpoint.
`destAddr` – The destination address.
`clusterID` – The identifier of the cluster.
`discoverRspCmd` - The Discover Response command to be sent.
`direction` - The client/server direction of the command.
`disableDefaultRsp` - Disable Default Response command.
`seqNum` - The identification number for the transaction.

#### 3.16.4 Return
`ZStatus_t` – enum found in ZComDef.h.

 

### 3.17 Register Attribute List

#### 3.17.1 Description
This function is used to register an Attribute List with ZCL Foundation.

#### 3.17.2 Prototype

```
ZStatus_t zcl_registerAttrList( uint8 endpoint, uint8 numAttr,
                                zclAttrRec_t *newAttrList );
```

#### 3.17.3 Parameter Details
`endpoint` – The endpoint the attribute list belongs to.
`numAttr` - The number of attributes in the list.
`newAttrList` – The array of attribute records.

#### 3.17.4 Return
`ZStatus_t` – enum found in ZComDef.h.

### 3.18 Register Attribute Data Validation Callback

#### 3.18.1 Description
This function is used to register an Attribute Data Validation Callback function with ZCL Foundation.

#### 3.18.2 Prototype

```
ZStatus_t zcl_registerVaildateAttrData(
                        zclValidateAttrData_t pfnValidateAttrData );
```

#### 3.18.3 Parameter Details
`pfnValidateAttrData` – The function pointer to the attribute data validation routine.

#### 3.18.4 Return
`ZStatus_t` – enum found in ZComDef.h.

### 3.19 Register Cluster Library Handler Callback

#### 3.19.1 Description
This function is used to register a Cluster Library Handler callback function with the ZCL Foundation layer.

#### 3.19.2 Prototype

```
ZStatus_t zcl_registerPlugin( uint16 startClusterID,
                              uint16 endClusterID,
                              zclInHdlr_t pfnIncomingHdlr );
```

#### 3.19.3 Parameter Details
`startClusterID` – The starting cluster ID.
`endClusterID` - The ending cluster ID.
`pfnIncomingHdlr` – The function pointer to incoming message handler.

#### 3.19.4 Return
`ZStatus_t` – enum found in ZComDef.h.

 

## 3.20 Cluster Library Handler Callback

### 3.20.1 Description
This callback function is called to handle an incoming cluster-specific message from ZCL Foundation.

### 3.20.2 Prototype

```
typedef ZStatus_t (*zclInHdlr_t)( zclIncoming_t *msg );
```

### 3.20.3 Parameter Details
`msg` – The incoming message.

### 3.20.4 Return
`ZStatus_t` – enum found in ZComDef.h.

## 3.21 Register Cluster Option List

### 3.21.1 Description
This function is used to register a Cluster Option List the ZCL Foundation layer. This API should be called to enable 'Application Link Key' security and/or 'APS ACK' or a specific Cluster. The 'Application Link Key' is discarded if security isn't enabled on the device. The default behavior is 'Network Key' when security is enabled and no 'APS ACK' for the ZCL messages.

### 3.21.2 Prototype

```
ZStatus_t zcl_registerClusterOptionList( uint8 endpoint,
                                         uint8 numOption,
                                         zclOptionRec_t optionList[] );
```

### 3.21.3 Parameter Details
`endpoint` – The endpoint the option list belongs to.
`numOption` – The number of options in the list.
`optionList` – Array of cluster option records.

### 3.21.4 Return
`ZStatus_t` – enum found in ZComDef.h.

## 3.22 Get the Raw AF Incoming Message

### 3.22.1 Description
This function is used during a callback function to retrieve a pointer to the raw AF incoming message. This function can only be called during a ZCL callback function and the calling function MUST not change any data in the message.

### 3.22.2 Prototype

```
afIncomingMSGPacket_t *zcl_getRawAFMsg( void );
```

### 3.22.3 Parameter Details
`None`.

### 3.22.4 Return
Returns a pointer to the original AF message or NULL if not currently processing an AF message.

 

# 4. General Functional Domain

## 4.1 Introduction

The General functional domain contains the following clusters:

- Basic
- Power Configuration
- Device Temperature Configuration
- Identity
- Groups
- Scenes
- On/Off
- On/Off Switch Configuration
- Level Control
- Alarms
- Time
- RSSI Indication

The Basic, Identity, Groups, Scenes, On/Off, Level Control, Alarms and RSSI Indication clusters provide commands but the Power Configuration, Device Temperature Configuration, On/Off Switch Configuration and Time clusters don't provide any commands.

## 4.2 Send Reset to Factory Defaults (Basic)

### 4.2.1 Description

This function is used to send a Reset to Factory Defaults Command.

### 4.2.2 Prototype

```
ZStatus_t zclGeneral_SendBasicResetFactoryDefaults( uint8 srcEP,
                                                    afAddrType_t *dstAddr,
                                                    uint8 disableDefaultRsp,
                                                    uint8 seqNum );
```

### 4.2.3 Parameter Details

`srcEP` – The source endpoint.
`destAddr` – The destination address.
`disableDefaultRsp` - Disable Default Response command.
`seqNum` - The identification number for the transaction.

### 4.2.4 Return

`ZStatus_t` – enum found in ZComDef.h.

 

### 4.3 Send Identify (Identify)

#### 4.3.1 Description
This function is used to send an Identify command.

#### 4.3.2 Prototype

```
ZStatus_t zclGeneral_SendIdentify( uint8 srcEP,
                                   afAddrType_t *dstAddr,
                                   uint8 disableDefaultRsp, uint8 seqNum );
```

#### 4.3.3 Parameter Details
`srcEP` – The source endpoint.
`destAddr` – The destination address.
`disableDefaultRsp` - Disable Default Response command.
`seqNum` - The identification number for the transaction.

#### 4.3.4 Return
`ZStatus_t` – enum found in ZComDef.h.

### 4.4 Send Identify Query (Identify)

#### 4.4.1 Description
This function is used to send an Identify Query command.

#### 4.4.2 Prototype

```
ZStatus_t zclGeneral_SendIdentifyQuery( uint8 srcEP,
                                        afAddrType_t *dstAddr,
                                        uint8 disableDefaultRsp, uint8 seqNum );
```

#### 4.4.3 Parameter Details
`srcEP` – The source endpoint.
`destAddr` – The destination address.
`disableDefaultRsp` - Disable Default Response command.
`seqNum` - The identification number for the transaction.

#### 4.4.4 Return
`ZStatus_t` – enum found in ZComDef.h.

 

## 4.5 Send Identify Query Response (Identify)

### 4.5.1 Description
This function is used to send an Identify Query Response command.

### 4.5.2 Prototype

```
ZStatus_t zclGeneral_SendIdentifyQueryResponse( uint8 srcEP,
                                    afAddrType_t *dstAddr, uint16 timeout,
                                    uint8 disableDefaultRsp, uint8 seqNum );
```

### 4.5.3 Parameter Details
`srcEP` – The source endpoint.
`destAddr` – The destination address.
`timeout` – How long the device will continue to identify itself (in seconds).
`disableDefaultRsp` - Disable Default Response command.
`seqNum` - The identification number for the transaction.

### 4.5.4 Return
`ZStatus_t` – enum found in ZComDef.h.

## 4.6 Send Add Group (Group)

### 4.6.1 Description
This function is used to send an Add Group command.

### 4.6.2 Prototype

```
ZStatus_t zclGeneral_SendGroupAdd( uint8 srcEP, afAddrType_t *dstAddr,
                                   int16 groupID, uint8 *groupName,
                                   uint8 disableDefaultRsp, uint8 seqNum );
```

### 4.6.3 Parameter Details
`srcEP` – The source endpoint.
`destAddr` – The destination address.
`groupID` – The ID of the Group to be added
`groupName` – The name of the Group to be added
`disableDefaultRsp` - Disable Default Response command.
`seqNum` - The identification number for the transaction.

### 4.6.4 Return
`ZStatus_t` – enum found in ZComDef.h.

 

## 4.7 Send View Group (Group)

### 4.7.1 Description
This function is used to send a View Group command.

### 4.7.2 Prototype

```
ZStatus_t zclGeneral_SendGroupView( uint8 srcEP, afAddrType_t *dstAddr,
                                    int16 groupID, uint8 disableDefaultRsp,
                                    uint8 seqNum );
```

### 4.7.3 Parameter Details
`srcEP` – The source endpoint.
`destAddr` – The destination address.
`groupID` – The ID of the Group to be viewed
`disableDefaultRsp` - Disable Default Response command.
`seqNum` - The identification number for the transaction.

### 4.7.4 Return
`ZStatus_t` – enum found in ZComDef.h.

## 4.8 Send Get Group Membership (Group)

### 4.8.1 Description
This function is used to send a Get Group Membership command.

### 4.8.2 Prototype

```
ZStatus_t zclGeneral_ SendGroupGetMembership( uint8 srcEP,
                                    afAddrType_t *dstAddr, uint8 grpCnt,
                                    uint16 *grpList, uint8 disableDefaultRsp,
                                    uint8 seqNum );
```

### 4.8.3 Parameter Details
`srcEP` – The source endpoint.
`destAddr` – The destination address.
`grpCnt` – The number of the groups in the Group list
`grpList` – The Group IDs of which the entity is a member
`disableDefaultRsp` - Disable Default Response command.
`seqNum` - The identification number for the transaction.

### 4.8.4 Return
`ZStatus_t` – enum found in ZComDef.h.

 

### 4.9 Send Remove Group (Group)

#### 4.9.1 Description
This function is used to send a Remove Group command.

#### 4.9.2 Prototype

```
ZStatus_t zclGeneral_SendGroupRemove( uint8 srcEP, afAddrType_t *dstAddr,
                                      uint16 *groupID, uint8 disableDefaultRsp,
                                      uint8 seqNum );
```

#### 4.9.3 Parameter Details
`srcEP` – The source endpoint.
`destAddr` – The destination address.
`grpList` – The ID of the Group to be removed.
`disableDefaultRsp` - Disable Default Response command.
`seqNum` - The identification number for the transaction.

#### 4.9.4 Return
`ZStatus_t` – enum found in ZComDef.h.

### 4.10 Send Remove All Groups (Group)

#### 4.10.1 Description
This function is used to send a Remove All Groups command.

#### 4.10.2 Prototype

```
ZStatus_t zclGeneral_SendGroupRemoveAll( uint8 srcEP, afAddrType_t *dstAddr,
                                         uint8 disableDefaultRsp, uint8 seqNum );
```

#### 4.10.3 Parameter Details
`srcEP` – The source endpoint.
`destAddr` – The destination address.
`disableDefaultRsp` - Disable Default Response command.
`seqNum` - The identification number for the transaction.

#### 4.10.4 Return
`ZStatus_t` – enum found in ZComDef.h.

 

### 4.11 Send Add Group If Identifying (Group)

#### 4.11.1 Description
This function is used to send an Add Group If Identifying command.

#### 4.11.2 Prototype

```
ZStatus_t zclGeneral_ SendGroupAddIfIdentifying( uint8 srcEP,
                                    afAddrType_t *dstAddr, uint16 groupID,
                                    uint8 *groupName, uint8 disableDefaultRsp,
                                    uint8 seqNum );
```

#### 4.11.3 Parameter Details
`srcEP` – The source endpoint.
`destAddr` – The destination address.
`groupID` – The ID of the Group to be added.
`disableDefaultRsp` - Disable Default Response command.
`groupName` – The name of the Group to be added.
`seqNum` - The identification number for the transaction.

#### 4.11.4 Return
`ZStatus_t` – enum found in ZComDef.h.

### 4.12 Send Add Group Response (Group)

#### 4.12.1 Description
This function is used to send an Add Group Response command.

#### 4.12.2 Prototype

```
ZStatus_t zclGeneral_SendGroupAddResponse( uint8 srcEP,
                                     afAddrType_t *dstAddr, uint8 status,
                                     uint16 groupID, uint8 disableDefaultRsp,
                                     uint8 seqNum );
```

#### 4.12.3 Parameter Details
`srcEP` – The source endpoint.
`destAddr` – The destination address.
`status` – The status of the Group Add command.
`groupID` – The ID of the Group which is being added.
`disableDefaultRsp` - Disable Default Response command.
`seqNum` - The identification number for the transaction.

#### 4.12.4 Return
`ZStatus_t` – enum found in ZComDef.h.

### 4.13 Send View Group Response (Group)

#### 4.13.1 Description
This function is used to send a View Group Response command.

#### 4.13.2 Prototype

 

```
ZStatus_t zclGeneral_ SendGroupViewResponse( uint8 srcEP,
                                    afAddrType_t *dstAddr, uint8 status,
                                    aps_Group_t *grp, uint8 disableDefaultRsp,
                                    uint8 seqNum );
```

#### 4.13.3 Parameter Details

srcEP – The source endpoint.
destAddr – The destination address.
status – The status of the Group View command.
grp – The Group info to be viewed.
disableDefaultRsp - Disable Default Response command.
seqNum - The identification number for the transaction.

#### 4.13.4 Return

ZStatus_t – enum found in ZComDef.h.

### 4.14 Send Get Group Membership Response (Group)

#### 4.14.1 Description

This function is used to send a Get Group Membership Response command.

#### 4.14.2 Prototype

```
ZStatus_t zclGeneral_ SendGroupGetMembershipResponse( uint8 srcEP,
                                afAddrType_t *dstAddr, uint8 capacity,
                                uint8 grpCnt, uint16 *grpList,
                                uint8 disableDefaultRsp, uint8 seqNum );
```

#### 4.14.3 Parameter Details

srcEP – The source endpoint.
destAddr – The destination address.
capacity – The remaining capacity of the Group table of the device.
grpCnt – The number of groups contained in the Group List field.
grpList – The IDs either of all the groups in the Group table (if Group List of Get Group Membership was empty) or all the groups from the Group List of Get Group Membership command which are in the Group table.
disableDefaultRsp - Disable Default Response command.
seqNum - The identification number for the transaction.

#### 4.14.4 Return

ZStatus_t – enum found in ZComDef.h.

 

### 4.15 Send Remove Group Response (Group)

#### 4.15.1 Description
This function is used to send a Remove Group Response command.

#### 4.15.2 Prototype

```
ZStatus_t zclGeneral_ SendGroupRemoveResponse( uint8 srcEP,
                                afAddrType_t *dstAddr, uint8 status,
                                uint16 groupID, uint8 disableDefaultRsp,
                                uint8 seqNum );
```

#### 4.15.3 Parameter Details
`srcEP` – The source endpoint.
`destAddr` – The destination address.
`status` – The status of the Remove Group command.
`groupID` – The ID of the Group that was to be removed.
`disableDefaultRsp` - Disable Default Response command.
`seqNum` - The identification number for the transaction.

#### 4.15.4 Return
`ZStatus_t` – enum found in ZComDef.h.

### 4.16 Send Add Scene (Scene)

#### 4.16.1 Description
This function is used to send an Add Scene command.

#### 4.16.2 Prototype

```
ZStatus_t zclGeneral_SendSceneAdd( uint8 srcEP, afAddrType_t *dstAddr,
                                   zclGeneral_Scene_t *scene,
                                   uint8 disableDefaultRsp, uint8 seqNum );
```

#### 4.16.3 Parameter Details
`srcEP` – The source endpoint.
`destAddr` – The destination address.
`scene` – The scene to be added.
`disableDefaultRsp` - Disable Default Response command.
`seqNum` - The identification number for the transaction.

#### 4.16.4 Return
`ZStatus_t` – enum found in ZComDef.h.

 

### 4.17 Send View Scene (Scene)

#### 4.17.1 Description
This function is used to send a View Scene command.

#### 4.17.2 Prototype

```
ZStatus_t zclGeneral_SendSceneView( uint8 srcEP, afAddrType_t *dstAddr,
                                    int16 groupID, uint8 sceneID,
                                    uint8 disableDefaultRsp, uint8 seqNum );
```

#### 4.17.3 Parameter Details
`srcEP` – The source endpoint.
`destAddr` – The destination address.
`groupID` – The ID of the Group that the Scene belongs to.
`sceneID` – The ID of the Scene to be viewed.
`disableDefaultRsp` - Disable Default Response command.
`seqNum` - The identification number for the transaction.

#### 4.17.4 Return
`ZStatus_t` – enum found in ZComDef.h.

### 4.18 Send Remove Scene (Scene)

#### 4.18.1 Description
This function is used to send a Remove Scene command.

#### 4.18.2 Prototype

```
ZStatus_t zclGeneral_ SendSceneRemove( uint8 srcEP, afAddrType_t *dstAddr,
                                    int16 groupID, uint8 sceneID,
                                    uint8 disableDefaultRsp, uint8 seqNum );
```

#### 4.18.3 Parameter Details
`srcEP` – The source endpoint.
`destAddr` – The destination address.
`groupID` – The ID of the Group that the Scene belongs to.
`sceneID` – The ID of the Scene to be removed.
`disableDefaultRsp` - Disable Default Response command.
`seqNum` - The identification number for the transaction.

#### 4.18.4 Return
`ZStatus_t` – enum found in ZComDef.h.

 

### 4.19 Send Remove All Scenes (Scene)

#### 4.19.1 Description
This function is used to send a Remove All Scenes command.

#### 4.19.2 Prototype

```
ZStatus_t zclGeneral_ SendSceneRemoveAll( uint8 srcEP, afAddrType_t *dstAddr,
                                          int16 groupID, uint8 disableDefaultRsp,
                                          uint8 seqNum );
```

#### 4.19.3 Parameter Details
`srcEP` – The source endpoint.
`destAddr` – The destination address.
`groupID` – The ID of the Group for which all the Scenes to be removed.
`disableDefaultRsp` - Disable Default Response command.
`seqNum` - The identification number for the transaction.

#### 4.19.4 Return
`ZStatus_t` – enum found in ZComDef.h.

### 4.20 Send Store Scene (Scene)

#### 4.20.1 Description
This function is used to send a Store Scene command.

#### 4.20.2 Prototype

```
ZStatus_t zclGeneral_SendSceneStore( uint8 srcEP, afAddrType_t *dstAddr,
                                     int16 groupID, uint8 sceneID,
                                     uint8 disableDefaultRsp, uint8 seqNum );
```

#### 4.20.3 Parameter Details
`srcEP` – The source endpoint.
`destAddr` – The destination address.
`groupID` – The ID of the Group that the Scene belongs to.
`sceneID` – The ID of the Scene to be stored.
`disableDefaultRsp` - Disable Default Response command.
`seqNum` - The identification number for the transaction.

#### 4.20.4 Return
`ZStatus_t` – enum found in ZComDef.h.

 

### 4.21 Send Recall Scene (Scene)

#### 4.21.1 Description
This function is used to send a Recall Scene command.

#### 4.21.2 Prototype

```
ZStatus_t zclGeneral_SendSceneRecall( uint8 srcEP, afAddrType_t *dstAddr,
                                      int16 groupID, uint8 sceneID,
                                      uint8 disableDefaultRsp, uint8 seqNum );
```

#### 4.21.3 Parameter Details
`srcEP` – The source endpoint.
`destAddr` – The destination address.
`groupID` – The ID of the Group that the Scene belongs to.
`sceneID` – The ID of the Scene to be recalled.
`disableDefaultRsp` - Disable Default Response command.
`seqNum` - The identification number for the transaction.

#### 4.21.4 Return
`ZStatus_t` – enum found in ZComDef.h.

### 4.22 Send Get Scene Membership (Scene)

#### 4.22.1 Description
This function is used to send a Get Scene Membership command.

#### 4.22.2 Prototype

```
ZStatus_t zclGeneral_ SendSceneGetMembership( uint8 srcEP,
                                      afAddrType_t *dstAddr, uint16 groupID,
                                      uint8 disableDefaultRsp, uint8 seqNum );
```

#### 4.22.3 Parameter Details
`srcEP` – The source endpoint.
`destAddr` – The destination address.
`groupID` – The Group ID of which the Scene is a member.
`disableDefaultRsp` - Disable Default Response command.
`seqNum` - The identification number for the transaction.

#### 4.22.4 Return
`ZStatus_t` – enum found in ZComDef.h.

 

### 4.23 Send Add Scene Response (Scene)

#### 4.23.1 Description
This function is used to send an Add Scene Response command.

#### 4.23.2 Prototype

```
ZStatus_t zclGeneral_SendSceneAddResponse( uint8 srcEP,
                                afAddrType_t *dstAddr, uint8 status,
                                uint16 groupID, uint8 sceneID,
                                uint8 disableDefaultRsp, uint8 seqNum );
```

#### 4.23.3 Parameter Details
`srcEP` – The source endpoint.
`destAddr` – The destination address.
`status` – The status of the Scene Add command.
`groupID` – The Group ID of the Scene which was added.
`sceneID` – The ID of the Scene which was added.
`disableDefaultRsp` - Disable Default Response command.
`seqNum` - The identification number for the transaction.

#### 4.23.4 Return
`ZStatus_t` – enum found in ZComDef.h.

### 4.24 Send View Scene Response (Scene)

#### 4.24.1 Description
This function is used to send a View Scene Response command.

#### 4.24.2 Prototype

```
ZStatus_t zclGeneral_SendSceneViewResponse( uint8 srcEP,
                                    afAddrType_t *dstAddr, uint8 status,
                                    zclGeneral_Scene_t *scene,
                                    uint8 disableDefaultRsp, uint8 seqNum );
```

#### 4.24.3 Parameter Details
`srcEP` – The source endpoint.
`destAddr` – The destination address.
`status` – The status of the Scene View command.
`scene` – The Scene info to be viewed.
`disableDefaultRsp` - Disable Default Response command.
`seqNum` - The identification number for the transaction.

#### 4.24.4 Return
`ZStatus_t` – enum found in ZComDef.h.

 

### 4.25 Send Remove Scene Response (Scene)

#### 4.25.1 Description
This function is used to send a Remove Scene Response command.

#### 4.25.2 Prototype

```
ZStatus_t zclGeneral_SendSceneRemove( uint8 srcEP, afAddrType_t *dstAddr,
                                      uint8 status, uint16 groupID, uint8 sceneID,
                                      uint8 disableDefaultRsp, uint8 seqNum );
```

#### 4.25.3 Parameter Details
`srcEP` – The source endpoint.
`destAddr` – The destination address.
`status` – The status of the Remove Scene command.
`groupID` – The Group ID of the Scene which was removed.
`sceneID` – The ID of the Scene which was removed.
`disableDefaultRsp` - Disable Default Response command.
`seqNum` - The identification number for the transaction.

#### 4.25.4 Return
`ZStatus_t` – enum found in ZComDef.h.

### 4.26 Send Remove All Scenes (Scene)

#### 4.26.1 Description
This function is used to send a Remove All Groups command.

#### 4.26.2 Prototype

```
ZStatus_t zclGeneral_SendSceneRemoveAll( uint8 srcEP, afAddrType_t *dstAddr,
                                         uint8 status, uint16 groupID,
                                         uint8 disableDefaultRsp, uint8 seqNum );
```

#### 4.26.3 Parameter Details
`srcEP` – The source endpoint.
`destAddr` – The destination address.
`status` – The status of the Remove All Scenes command.
`groupID` – The Group ID of the Scenes which were removed.
`disableDefaultRsp` - Disable Default Response command.
`seqNum` - The identification number for the transaction.

#### 4.26.4 Return
`ZStatus_t` – enum found in ZComDef.h.

 

### 4.27 Send Store Scene (Scene)

#### 4.27.1 Description
This function is used to send a Store Scene command.

#### 4.27.2 Prototype

```
ZStatus_t zclGeneral_SendSceneRemove( uint8 srcEP, afAddrType_t *dstAddr,
                                      uint8 status, uint16 groupID,
                                      uint8 sceneID, uint8 disableDefaultRsp,
                                      uint8 seqNum );
```

#### 4.27.3 Parameter Details
`srcEP` – The source endpoint.
`destAddr` – The destination address.
`status` – The status of the Store Scene command.
`groupID` – The Group ID of the Scene which was stored.
`sceneID` – The ID of the Scene which was stored.
`disableDefaultRsp` - Disable Default Response command.
`seqNum` - The identification number for the transaction.

#### 4.27.4 Return
`ZStatus_t` – enum found in ZComDef.h.

### 4.28 Send Get Scene Membership Response (Scene)

#### 4.28.1 Description
This function is used to send a Get Scene Membership Response command.

#### 4.28.2 Prototype

```
ZStatus_t zclGeneral_SendSceneGetMembershipResponse( uint8 srcEP,
                            afAddrType_t *dstAddr, uint8 status,
                            uint8 capacity, uint8 sceneCnt, uint8 *sceneList,
                            uint16 groupID, uint8 disableDefaultRsp,
                            uint8 seqNum );
```

#### 4.28.3 Parameter Details
`srcEP` – The source endpoint.
`destAddr` – The destination address.
`status` – The status of the Get Scene Membership command.
`capacity` – The remaining capacity of the Scene table of the device.
`sceneCnt` – The number of scenes contained in the Scene List field.
`sceneList` – The IDs of all the scenes in the Scene table with the corresponding Group ID.
`groupID` – The Group ID of the Scenes.
`disableDefaultRsp` - Disable Default Response command.
`seqNum` - The identification number for the transaction.

#### 4.28.4 Return
`ZStatus_t` – enum found in ZComDef.h.

 

### 4.29 Send Off (On/Off)

#### 4.29.1 Description
This function is used to send an Off command.

#### 4.29.2 Prototype

```
ZStatus_t zclGeneral_SendOnOff_CmdOff( uint8 srcEP, afAddrType_t *dstAddr,
                                       uint8 disableDefaultRsp, uint8 seqNum );
```

#### 4.29.3 Parameter Details
`srcEP` – The source endpoint.
`destAddr` – The destination address.
`disableDefaultRsp` - Disable Default Response command.
`seqNum` - The identification number for the transaction.

#### 4.29.4 Return
`ZStatus_t` – enum found in ZComDef.h.

### 4.30 Send On (On/Off)

#### 4.30.1 Description
This function is used to send an On command.

#### 4.30.2 Prototype

```
ZStatus_t zclGeneral_SendOnOff_CmdOn( uint8 srcEP, afAddrType_t *dstAddr,
                                      uint8 disableDefaultRsp, uint8 seqNum );
```

#### 4.30.3 Parameter Details
`srcEP` – The source endpoint.
`destAddr` – The destination address.
`disableDefaultRsp` - Disable Default Response command.
`seqNum` - The identification number for the transaction.

#### 4.30.4 Return
`ZStatus_t` – enum found in ZComDef.h.

 

### 4.31 Send Toggle (On/Off)

#### 4.31.1 Description
This function is used to send a Toggle command.

#### 4.31.2 Prototype

```
ZStatus_t zclGeneral_SendOnOff_CmdToggle( uint8 srcEP,
                                          afAddrType_t *dstAddr,
                                          uint8 disableDefaultRsp, uint8 seqNum );
```

#### 4.31.3 Parameter Details
`srcEP` – The source endpoint.
`destAddr` – The destination address.
`disableDefaultRsp` - Disable Default Response command.
`seqNum` - The identification number for the transaction.

#### 4.31.4 Return
`ZStatus_t` – enum found in ZComDef.h.

### 4.32 Send Move to Level (Level Control)

#### 4.32.1 Description
This function is used to send a Move to Level command.

#### 4.32.2 Prototype

```
ZStatus_t zclGeneral_SendLevelControlMoveToLevel( uint8 srcEP,
                         afAddrType_t *dstAddr, uint8 level, uint16 transTime,
                         uint8 disableDefaultRsp, uint8 seqNum );
```

#### 4.32.3 Parameter Details
`srcEP` – The source endpoint.
`destAddr` – The destination address.
`level` – The new level to move to.
`transTime` – The time (in seconds) that shall be taken to move to the new level.
`disableDefaultRsp` - Disable Default Response command.
`seqNum` - The identification number for the transaction.

#### 4.32.4 Return
`ZStatus_t` – enum found in ZComDef.h.

 

### 4.33 Send Move (Level Control)

#### 4.33.1 Description
This function is used to send a Move command.

#### 4.33.2 Prototype

```
ZStatus_t zclGeneral_SendLevelControlMove( uint8 srcEP,
                                           afAddrType_t *dstAddr, uint8 moveMode,
                                           uint8 rate, uint8 disableDefaultRsp,
                                           uint8 seqNum );
```

#### 4.33.3 Parameter Details
`srcEP` – The source endpoint.
`destAddr` – The destination address.
`moveMode` – The move mode.
`rate` – The rate of movement in steps per second.
`disableDefaultRsp` - Disable Default Response command.
`seqNum` - The identification number for the transaction.

#### 4.33.4 Return
`ZStatus_t` – enum found in ZComDef.h.

### 4.34 Send Step (Level Control)

#### 4.34.1 Description
This function is used to send a Step command.

#### 4.34.2 Prototype

```
ZStatus_t zclGeneral_SendLevelControlStep( uint8 srcEP,
                                afAddrType_t *dstAddr, uint8 stepMode,
                                uint8 amount, uint16 transTime,
                                uint8 disableDefaultRsp, uint8 seqNum );
```

#### 4.34.3 Parameter Details
`srcEP` – The source endpoint.
`destAddr` – The destination address.
`stepMode` – The step mode.
`amount` – The number of levels to step.
`transTime` – The time (in 1/10ths of seconds) that shall be taken to perform the step.
`disableDefaultRsp` - Disable Default Response command.
`seqNum` - The identification number for the transaction.

#### 4.34.4 Return
`ZStatus_t` – enum found in ZComDef.h.

 

### 4.35 Send Reset Alarm (Alarm)

#### 4.35.1 Description
This function is used to send a Reset Alarm command.

#### 4.35.2 Prototype

```
ZStatus_t zclGeneral_SendAlarmReset( uint8 srcEP, afAddrType_t *dstAddr,
                                     uint8 alarmCode, uint16 clusterID,
                                     uint8 disableDefaultRsp, uint8 seqNum );
```

#### 4.35.3 Parameter Details
`srcEP` – The source endpoint.
`destAddr` – The destination address.
`alarmCode` – The identifying Code for the cause of the alarm.
`clusterID` – The Identifier of the Cluster whose attribute generated the alarm.
`disableDefaultRsp` - Disable Default Response command.
`seqNum` - The identification number for the transaction.

#### 4.35.4 Return
`ZStatus_t` – enum found in ZComDef.h.

### 4.36 Send Reset All Alarms (Alarm)

#### 4.36.1 Description
This function is used to send a Reset All Alarms command.

#### 4.36.2 Prototype

```
ZStatus_t zclGeneral_SendAlarmResetAll( uint8 srcEP, afAddrType_t *dstAddr,
                                        uint8 disableDefaultRsp, uint8 seqNum );
```

#### 4.36.3 Parameter Details
`srcEP` – The source endpoint.
`destAddr` – The destination address.
`disableDefaultRsp` - Disable Default Response command.
`seqNum` - The identification number for the transaction.

#### 4.36.4 Return
`ZStatus_t` – enum found in ZComDef.h.

 

### 4.37 Send Get Alarm (Alarm)

#### 4.37.1 Description
This function is used to send a Get Alarm command.

#### 4.37.2 Prototype

```
ZStatus_t zclGeneral_ SendAlarmGet( uint8 srcEP, afAddrType_t *dstAddr,
                                    uint8 disableDefaultRsp, uint8 seqNum );
```

#### 4.37.3 Parameter Details
`srcEP` – The source endpoint.
`destAddr` – The destination address.
`disableDefaultRsp` - Disable Default Response command.
`seqNum` - The identification number for the transaction.

#### 4.37.4 Return
`ZStatus_t` – enum found in ZComDef.h.

### 4.38 Send Reset Alarm Log (Alarm)

#### 4.38.1 Description
This function is used to send a Reset Alarm Log command.

#### 4.38.2 Prototype

```
ZStatus_t zclGeneral_ SendAlarmResetLog( uint8 srcEP, afAddrType_t *dstAddr,
                                         uint8 disableDefaultRsp, uint8 seqNum );
```

#### 4.38.3 Parameter Details
`srcEP` – The source endpoint.
`destAddr` – The destination address.
`disableDefaultRsp` - Disable Default Response command.
`seqNum` - The identification number for the transaction.

#### 4.38.4 Return
`ZStatus_t` – enum found in ZComDef.h.

 

### 4.39 Send Alarm (Alarm)

#### 4.39.1 Description
This function is used to send an Alarm command.

#### 4.39.2 Prototype

```
ZStatus_t zclGeneral_ SendAlarm( uint8 srcEP, afAddrType_t *dstAddr,
                                 uint8 alarmCode, uint16 clusterID,
                                 uint8 disableDefaultRsp, uint8 seqNum );
```

#### 4.39.3 Parameter Details
`srcEP` – The source endpoint.
`destAddr` – The destination address.
`alarmCode` – The identifying Code for the cause of the alarm.
`clusterID` – The Identifier of the Cluster whose attribute generated the alarm.
`disableDefaultRsp` - Disable Default Response command.
`seqNum` - The identification number for the transaction.

#### 4.39.4 Return
`ZStatus_t` – enum found in ZComDef.h.

### 4.40 Send Get Alarm Response (Alarm)

#### 4.40.1 Description
This function is used to send a Get Alarm Response command.

#### 4.40.2 Prototype

```
ZStatus_t zclGeneral_SendAlarmGetRespnose( uint8 srcEP,
                                    afAddrType_t *dstAddr, uint8 status,
                                    uint8 alarmCode, uint16 clusterID,
                                    uint32 timeStamp, uint8 disableDefaultRsp,
                                    uint8 seqNum );
```

#### 4.40.3 Parameter Details
`srcEP` – The source endpoint.
`destAddr` – The destination address.
`status` – The status of the Get Alarm command.
`alarmCode` – The identifying Code for the cause of the alarm.
`timeStamp` – The time at which the alarm occurred.
`clusterID` – The Identifier of the Cluster whose attribute generated the alarm.
`disableDefaultRsp` - Disable Default Response command.
`seqNum` - The identification number for the transaction.

#### 4.40.4 Return
`ZStatus_t` – enum found in ZComDef.h.

 

### 4.41 Send Set Absolute Location (RSSI Location)

#### 4.41.1 Description
This function is used to send a Set Absolute Location command.

#### 4.41.2 Prototype

```
ZStatus_t zclGeneral_SendLocationSetAbsolute( uint8 srcEP,
                                              afAddrType_t *dstAddr,
                                              zclLocationAbsolute_t *absLoc,
                                              uint8 disableDefaultRsp,
                                              uint8 seqNum );
```

#### 4.41.3 Parameter Details
`srcEP` – The source endpoint.
`destAddr` – The destination address.
`absLoc` – The Absolute Location info.
`disableDefaultRsp` - Disable Default Response command.
`seqNum` - The identification number for the transaction.

#### 4.41.4 Return
`ZStatus_t` – enum found in ZComDef.h.

### 4.42 Send Set Device Configuration (RSSI Location)

#### 4.42.1 Description
This function is used to send a Set Device Configuration command.

#### 4.42.2 Prototype

```
ZStatus_t zclGeneral_SendLocationSetDevCfg( uint8 srcEP,
                                            afAddrType_t *dstAddr,
                                            zclLocationDevCfg_t *devCfg,
                                            uint8 disableDefaultRsp,
                                            uint8 seqNum );
```

#### 4.42.3 Parameter Details
`srcEP` – The source endpoint.
`destAddr` – The destination address.
`devCfg` – The Device Configuration info.
`disableDefaultRsp` - Disable Default Response command.
`seqNum` - The identification number for the transaction.

#### 4.42.4 Return
`ZStatus_t` – enum found in ZComDef.h.

 

### 4.43 Send Get Device Configuration (RSSI Location)

#### 4.43.1 Description
This function is used to send a Get Device Configuration command.

#### 4.43.2 Prototype

```
ZStatus_t zclGeneral_SendLocationGetDevCfg( uint8 srcEP,
                                            afAddrType_t *dstAddr,
                                            uint8 *targetAddr,
                                            uint8 disableDefaultRsp,
                                            uint8 seqNum );
```

#### 4.43.3 Parameter Details
`srcEP` – The source endpoint.
`destAddr` - The destination address.
`targetAddr` – The 64-bit IEEE Address of the device for which the location parameters are being requested.
`disableDefaultRsp` - Disable Default Response command.
`seqNum` - The identification number for the transaction.

#### 4.43.4 Return
`ZStatus_t` – enum found in ZComDef.h.

### 4.44 Send Get Location Data (RSSI Location)

#### 4.44.1 Description
This function is used to send a Get Location Data command.

#### 4.44.2 Prototype

```
ZStatus_t zclGeneral_ SendLocationGetData( uint8 srcEP,
                                           afAddrType_t *dstAddr,
                                           zclLocationGetData_t *locData,
                                           uint8 disableDefaultRsp,
                                           uint8 seqNum );
```

#### 4.44.3 Parameter Details
`srcEP` – The source endpoint.
`destAddr` - The destination address.
`locData` – Device's location information and channel parameters that are requested.
`disableDefaultRsp` - Disable Default Response command.
`seqNum` - The identification number for the transaction.

#### 4.44.4 Return
`ZStatus_t` – enum found in ZComDef.h.

 

### 4.45 Send Device Configuration Response (RSSI Location)

#### 4.45.1 Description
This function is used to send a Device Configuration Response command.

#### 4.45.2 Prototype

```
ZStatus_t zclGeneral_SendLocationDevCfgResponse( uint8 srcEP,
                                                 afAddrType_t *dstAddr,
                                                 zclLocationDevCfgRsp_t *devCfg,
                                                 uint8 disableDefaultRsp,
                                                 uint8 seqNum );
```

#### 4.45.3 Parameter Details
`srcEP` – The source endpoint.
`destAddr` - The destination address.
`devCfg` – The device's location parameters that are requested.
`disableDefaultRsp` - Disable Default Response command.
`seqNum` - The identification number for the transaction.

#### 4.45.4 Return
`ZStatus_t` – enum found in ZComDef.h.

### 4.46 Send Location Data Response (RSSI Location)

#### 4.46.1 Description
This function is used to send a Location Data Response command.

#### 4.46.2 Prototype

```
ZStatus_t zclGeneral_SendLocationDataResponse( uint8 srcEP,
                                               afAddrType_t *dstAddr,
                                               zclLocationDataRsp_t *locData,
                                               uint8 disableDefaultRsp,
                                               uint8 seqNum );
```

#### 4.46.3 Parameter Details
`srcEP` – The source endpoint.
`destAddr` - The destination address.
`locData` – Device's location information and channel parameters that are requested.
`disableDefaultRsp` - Disable Default Response command.
`seqNum` - The identification number for the transaction.

#### 4.46.4 Return
`ZStatus_t` – enum found in ZComDef.h.

 

### 4.47 Send Location Data Notification (RSSI Location)

#### 4.47.1 Description
This function is used to send a Location Data Notification command.

#### 4.47.2 Prototype

```
ZStatus_t zclGeneral_SendLocationData Notif( uint8 srcEP,
                                             afAddrType_t *dstAddr,
                                             zclLocationData_t *locData,
                                             uint8 disableDefaultRsp,
                                             uint8 seqNum );
```

#### 4.47.3 Parameter Details
`srcEP` – The source endpoint.
`destAddr` - The destination address.
`locData` – Device's location information and channel parameters that are requested.
`disableDefaultRsp` - Disable Default Response command.
`seqNum` - The identification number for the transaction.

#### 4.47.4 Return
`ZStatus_t` – enum found in ZComDef.h.

### 4.48 Send Compact Location Data Notification (RSSI Location)

#### 4.48.1 Description
This function is used to send a Compact Location Data Notification command.

#### 4.48.2 Prototype

```
ZStatus_t zclGeneral_SendLocationDataCompactNotif( uint8 srcEP,
                                                   afAddrType_t *dstAddr,
                                                   zclLocationData_t *locData,
                                                   uint8 disableDefaultRsp,
                                                   uint8 seqNum );
```

#### 4.48.3 Parameter Details
`srcEP` – The source endpoint.
`destAddr` - The destination address.
`locData` – Device's location information and channel parameters that are requested.
`disableDefaultRsp` - Disable Default Response command.
`seqNum` - The identification number for the transaction.

#### 4.48.4 Return
`ZStatus_t` – enum found in ZComDef.h.

 

### 4.49 Send RSSI Ping (RSSI Location)

#### 4.49.1 Description
This function is used to send a RSSI Ping command.

#### 4.49.2 Prototype

```
ZStatus_t zclGeneral_ SendRSSIPing( uint8 srcEP, afAddrType_t *dstAddr,
                                    uint8 locationType,
                                    uint8 disableDefaultRsp, uint8 seqNum );
```

#### 4.49.3 Parameter Details
`srcEP` – The source endpoint.
`destAddr` – The destination address.
`locationType` – The device's Location Type.
`disableDefaultRsp` - Disable Default Response command.
`seqNum` - The identification number for the transaction.

#### 4.49.4 Return
`ZStatus_t` – enum found in ZComDef.h.

### 4.50 Attribute Data Validation Callback

#### 4.50.1 Description
This callback is called to see if the supplied value for the attribute data is within the specified range of the attribute.

#### 4.50.2 Prototype

```
typedef uint8 (*zclValidateAttrData_t)( zclAttrRec_t *pAttr,
                                        zclWriteRec_t *pAttrInfo );
```

#### 4.50.3 Parameter Details
`pAttr` – Where data to be written.
`pAttrInfo` – Pointer to the attribute id, type and data.

#### 4.50.4 Return
`Uint8` – TRUE if the attribute data is valid. FALSE otherwise.

 

### 4.51 Register Application Command Callback

#### 4.51.1 Description
This function is used to register an Application's Command callbacks with the General functional domain.

#### 4.51.2 Prototype

```
ZStatus_t zclGeneral_RegisterCmdCallbacks( uint8 endpoint,
                                           zclGeneral_AppCallbacks_t *callbacks);
```

#### 4.51.3 Parameter Details
`endpoint` – The application's endpoint.
`callbacks` - Pointer to the callback records.

#### 4.51.4 Return
`ZStatus_t` – enum found in ZComDef.h.

### 4.52 Reset to Factory Defaults Callback

#### 4.52.1 Description
This callback is called to process an incoming Reset to Factory Defaults command. On receipt of this command, the device resets all the attributes of all its clusters to their factory defaults.

#### 4.52.2 Prototype

```
typedef void (*zclGCB_BasicReset_t)( void );
```

#### 4.52.3 Parameter Details
None.

#### 4.52.4 Return
None.

### 4.53 Identify Callback

#### 4.53.1 Description
This callback is called to process an incoming Identify command.

#### 4.53.2 Prototype

```
typedef void (*zclGCB_Identify_t)( zclIdentify_t *pCmd );
```

#### 4.53.3 Parameter Details
`pCmd` – received Identify command, which has the following fields:
`srcAddr` – The requestor's address.
`identifyTime` – The number of seconds the device shall continue to identify itself.

#### 4.53.4 Return
None.

 

### 4.54 Identify Response Callback

#### 4.54.1 Description
This callback is called to process an incoming Identify Response command.

#### 4.54.2 Prototype

```
typedef void (*zclGCB_IdentifyRsp_t)( zclIdentifyQueryRsp_t *pRsp );
```

#### 4.54.3 Parameter Details
`pRsp` – received Identify response, which has the following fields:
`srcAddr` – The requestor's address.
`timeout` – The number of seconds the device will continue to identify itself.

#### 4.54.4 Return
None.

### 4.55 On/Off/Toggle Callback

#### 4.55.1 Description
This callback is called to process an incoming On, Off or Toggle command.

#### 4.55.2 Prototype

```
typedef void (*zclGCB_OnOff_t)( uint8 cmd );
```

#### 4.55.3 Parameter Details
`cmd` – received command, which is either COMMAND_ON, COMMAND_OFF or COMMAND_TOGGLE.

#### 4.55.4 Return
None.

### 4.56 Move to Level Callback

#### 4.56.1 Description
This callback is called to process an incoming Level Control - Move to Level command.

#### 4.56.2 Prototype

```
typedef void (*zclGCB_LevelControlMoveToLevel_t)( zclLCMoveToLevel_t *pCmd );
```

#### 4.56.3 Parameter Details
`pCmd` – received Move to Level command, which has the following fields:
`level` – The new level to move to.
`transitionTime` – The time to take to move to the new level (in seconds).
`withOnOff` – With On/Off command .

#### 4.56.4 Return
None.

 

### 4.57 Move Callback

#### 4.57.1 Description
This callback is called to process an incoming Level Control - Move command.

#### 4.57.2 Prototype

```
typedef void (*zclGCB_LevelControlMove_t)( zclLCMove_t *pCmd );
```

#### 4.57.3 Parameter Details
`pCmd` – received Move command, which has the following fields:
`moveMode` – The move mode which is either LEVEL_MOVE_STOP, LEVEL_MOVE_UP, LEVEL_MOVE_ON_AND_UP, LEVEL_MOVE_DOWN, or LEVEL_MOVE_DOWN_AND_OFF.
`rate` – The rate of movement in steps per second.
`withOnOff` – With On/Off command.

#### 4.57.4 Return
None.

### 4.58 Step Callback

#### 4.58.1 Description
This callback is called to process an incoming Level Control - Step command.

#### 4.58.2 Prototype

```
typedef void (*zclGCB_LevelControlStep_t)( zclLCStep_t *pCmd );
```

#### 4.58.3 Parameter Details
`pCmd` – received Step command, which has the following fields:
`stepMode` – The step mode which is either LEVEL_STEP_UP, LEVEL_STEP_ON_AND_UP, LEVEL_STEP_DOWN, or LEVEL_STEP_DOWN_AND_OFF.
`amount` – The number of levels to step.
`transitionTime` – The time, in 1/10ths of a second, to take to perform the step.
`withOnOff` – With On/Off command .

#### 4.58.4 Return
None.

### 4.59 Step Callback

#### 4.59.1 Description
This callback is called to process an incoming Level Control - Stop command.

#### 4.59.2 Prototype

```
typedef void (*zclGCB_LevelControlStop_t)( void );
```

#### 4.59.3 Parameter Details
None.

#### 4.59.4 Return
None.

 

### 4.60 Group Response Callback

#### 4.60.1 Description
This callback is called to process an incoming Group Response command. This means that this application sent the request message.

#### 4.60.2 Prototype

```
typedef void (*zclGCB_GroupRsp_t)( zclGroupRsp_t *pRsp );
```

#### 4.60.3 Parameter Details
`pRsp` – received Group response, which has the following fields:
`srcAddr` – The requestor's address.
`cmdID` – The message ID which is either COMMAND_GROUP_ADD_RSP, COMMAND_GROUP_VIEW_RSP, COMMAND_GROUP_REMOVE_RSP or COMMAND_GROUP_GET_MEMBERSHIP_RSP.
`status` – The status which is either GROUP_STATUS_SUCCESS, GROUP_STATUS_TABLE_FULL, GROUP_STATUS_ALREADY_IN_TABLE, or GROUP_STATUS_NOT_IN_TABLE. Not valid for COMMAND_GROUP_GET_MEMBERSHIP_RSP.
`grpCnt` – The number of groups contained in the group list.
`grpList` – The group IDs that the action was performed on.
`capacity` – The remaining capacity of the group table.
`grpName` – The group name (only valid for COMMAND_GROUP_VIEW_RSP).

#### 4.60.4 Return
None.

### 4.61 Store Scene Callback

#### 4.61.1 Description
This callback is called to process an incoming Store Scene command. The application will fill in the "extField" with what is needed to restore its current settings.

#### 4.61.2 Prototype

```
typedef uint8 (*zclGCB_SceneStoreReq_t)( zclSceneReq_t *pReq );
```

#### 4.61.3 Parameter Details
`pReq` – received Scene Store request, which has the following fields:
`srcAddr` – The requestor's address.
`scene` – The scene information.

#### 4.61.4 Return
TRUE if extField is filled out, FALSE otherwise (in this case, there is no need to save the scene).

 

## 4.62 Recall Scene Callback

### 4.62.1 Description
This callback is called to process an incoming Recall Scene command. The application will use what's in the "extField" to restore to these settings.

### 4.62.2 Prototype

```
typedef void (*zclGCB_SceneRecallReq_t)( zclSceneReq_t *pReq );
```

### 4.62.3 Parameter Details
`pReq` – received Scene Recall request, which has the following fields:
`srcAddr` – The requestor's address.
`scene` – The scene information.

### 4.62.4 Return
None.

## 4.63 Scene Response Callback

### 4.63.1 Description
This callback is called to process an incoming Scene response message. This means that this application sent the request for this response.

### 4.63.2 Prototype

```
typedef void (*zclGCB_SceneRsp_t)( zclSceneRsp_t *pRsp );
```

### 4.63.3 Parameter Details
`pReq` – received Scene response, which has the following fields:
`srcAddr` – The requestor's address.
`cmdID` – The message ID which is either COMMAND_SCENE_ADD_RSP, COMMAND_SCENE_VIEW_RSP, COMMAND_SCENE_REMOVE_RSP, COMMAND_SCENE_REMOVE_ALL_RSP, COMMAND_SCENE_STORE_RSP or COMMAND_SCENE_GET_MEMBERSHIPSHIP_RSP.
`status` – The scene command status.
`sceneCnt` – The number of scenes contained in the scene list (only valid for COMMAND_SCENE_GET_MEMBERSHIPSHIP_RSP)
`sceneList` – The list of scene IDs (only valid for COMMAND_SCENE_GET_MEMBERSHIPSHIP_RSP)
`capacity` – The remaining capacity of the scene table (only valid for COMMAND_SCENE_GET_MEMBERSHIPSHIP_RSP)
`scene` – The scene information.

### 4.63.4 Return
None.

 

### 4.64 Alarm Callback

#### 4.64.1 Description
This callback is called to process an incoming Alarm request or response command.

#### 4.64.2 Prototype

```
typedef void (*zclGCB_Alarm_t)( zclAlarm_t *pAlarm );
```

#### 4.64.3 Parameter Details
`pReq` – received Alarm request, which has the following fields:
`srcAddr` – The requestor's address.
`cmdID` – The message ID which is either COMMAND_ALARMS_ALARM or COMMAND_ALARMS_GET_RSP
`status` – The alarm command status (only applicable to COMMAND_ALARMS_GET_RSP).
`alarmCode` – The identifying code for the cause of the alarm.
`clusterID` – The id of the cluster whose attribute generated this alarm
`timeStamp` – The time at which the alarm occurred (only applicable to COMMAND_ALARMS_GET_RSP)

#### 4.64.4 Return
None.

### 4.65 Location Callback

#### 4.65.1 Description
This callback is called to process an incoming RSSI Location command.

#### 4.65.2 Prototype

```
typedef void (*zclGCB_Location_t)( zclLocation_t *pCmd );
```

#### 4.65.3 Parameter Details
`pReq` – received Location command, which has the following fields:
`srcAddr` – The requestor's address.
`cmdID` – The message ID which is either COMMAND_LOCATION_SET_ABSOLUTE, COMMAND_LOCATION_SET_DEV_CFG, COMMAND_LOCATION_GET_DEV_CFG or COMMAND_LOCATION_GET_DATA.
`absLoc` – The Absolute Location info (only valid for COMMAND_LOCATION_SET_ABSOLUTE).
`loc` – The Location info (only valid for COMMAND_LOCATION_GET_DATA).
`devCfg` – The Device Configuration info (only valid for COMMAND_LOCATION_SET_DEV_CFG).
`ieeeAddr` – The device's IEEE Address (only valid for COMMAND_LOCATION_GET_DEV_CFG).
`seqNum` – The Sequence Number received with the message (only valid for GET commands).

#### 4.65.4 Return
None.

 

## 4.66 Location Response Callback

### 4.66.1 Description
This callback is called to process an incoming RSSI Location Response. This means that this application sent the request for this response.

### 4.66.2 Prototype

```
typedef void (*zclGCB_LocationRsp_t)( zclLocationRsp_t *pRsp );
```

### 4.66.3 Parameter Details
`pReq` – received Location response, which has the following fields:
`srcAddr` – The requestor's address.
`cmdID` – The message ID which is either COMMAND_LOCATION_DEV_CFG_RSP, COMMAND_LOCATION_DATA_RSP, COMMAND_LOCATION_DATA_NOTIF, COMMAND_LOCATION_COMPACT_DATA_NOTIF or COMMAND_LOCATION_RSSI_PING
`locRsp` – The Location Data Response command (applicable to all Data Response/Notification messages).
`devCfgRsp` – The Device Configuration Response command (only applicable to COMMAND_LOCATION_DEV_CFG_RSP).
`locationType` – The location type (only applicable to COMMAND_LOCATION_RSSI_PING).

### 4.66.4 Return
None.

## 4.67 Read Scene Count Callback

### 4.67.1 Description
This callback is used to read the number of scenes in stored in the local device's scene table (i.e., the Scene Count attribute). This function should be registered with the ZCL Foundation layer (using the zcl_registerReadWriteCB() API) when the data pointer '`dataPtr`' of the Scene Count attribute is set to NULL in the attribute database to be registered with ZCL. This callback function will be used by the ZCL task to process a ZCL Read Request operation on the Scene Count attribute.

### 4.67.2 Prototype

```
ZStatus_t zclGeneral_ReadSceneCountCB( uint16 clusterId, uint16 attrId,
                                       uint8 oper, uint8 *pValue,
                                       uint16 *pLen );
```

### 4.67.3 Parameter Details
`clusterId` – cluster that attribute belongs to
`attrId` - attribute to be read
`oper` - `ZCL_OPER_LEN` to get attribute length, or `ZCL_OPER_READ` to read attribute value
`pValue` - pointer to attribute value
`pLen` - pointer to length of attribute value read

### 4.67.4 Return
`ZCL_STATUS_SUCCESS` – Read operation was successful.
`ZCL_STATUS_SOFTWARE_FAILURE` – Read operation was unsuccessful.

 

# 5. Closures Functional Domain

## 5.1 Introduction

The Closures functional domain provides the following two clusters, defined by the Home Automation Public Application Profile:

- Door Lock Cluster
- Window Covering Cluster

The Door Lock Cluster provides an interface into a generic way to secure a door.
The Window Covering Cluster provides an interface for controlling and adjusting automatic window coverings such as drapery motors, automatic shades, and blinds.

## 5.2 Lock Door Command

### 5.2.1 Description

This command causes the lock device to lock the door.

### 5.2.2 Prototype

```
ZStatus_t zclClosures_SendDoorLock( uint8 srcEP,
      afAddrType_t *dstAddr,
      uint8 disableDefaultRsp,
      uint8 seqNum );
```

### 5.2.3 Parameter Details

`srcEP` – The source endpoint.
`destAddr` – The destination address.
`disableDefaultRsp` - Disable Default Response command.
`seqNum` - The identification number for the transaction.

### 5.2.4 Return

`ZStatus_t` – enum found in ZComDef.h.

## 5.3 Unlock Door Command

### 5.3.1 Description

This command causes the lock device to unlock the door.

### 5.3.2 Prototype

```
ZStatus_t zclClosures_SendDoorUnlock( uint8 srcEP,
  afAddrType_t *dstAddr,
  uint8 disableDefaultRsp,
  uint8 seqNum );
```

### 5.3.3 Parameter Details

`srcEP` – The source endpoint.
`destAddr` – The destination address.
`disableDefaultRsp` - Disable Default Response command.
`seqNum` - The identification number for the transaction.

### 5.3.4 Return

`ZStatus_t` – enum found in ZComDef.h.

 

## 5.4 Door Lock Response Command

### 5.4.1 Description
This command is sent in response to a lock command.

### 5.4.2 Prototype
```
ZStatus_t zclClosures_SendDoorLockRes( uint8 srcEP,
         afAddrType_t *dstAddr,
         uint8 disableDefaultRsp,
         uint8 seqNum,
         uint8 status );
```

### 5.4.3 Parameter Details
`srcEP` – The source endpoint.
`destAddr` – The destination address.
`disableDefaultRsp` - Disable Default Response command.
`seqNum` - The identification number for the transaction.
`Status` – The command status, either ZCL_SUCCESS or ZCL_FAILURE

### 5.4.4 Return
`ZStatus_t` – enum found in ZComDef.h.

## 5.5 Unlock Door Response

### 5.5.1 Description
This command is sent in response to an unlock door command.

### 5.5.2 Prototype
```
ZStatus_t zclClosures_SendDoorUnlockRes( uint8 srcEP,
     afAddrType_t *dstAddr,
     uint8 disableDefaultRsp,
     uint8 seqNum,
     uint8 status );
```

### 5.5.3 Parameter Details
`srcEP` – The source endpoint.
`destAddr` – The destination address.
`disableDefaultRsp` - Disable Default Response command.
`seqNum` - The identification number for the transaction.
`Status` – The command status, either ZCL_SUCCESS or ZCL_FAILURE

### 5.5.4 Return
`ZStatus_t` – enum found in ZComDef.h.

 

## 5.6 Up / Open Command (Window Covering)

### 5.6.1 Description
This function is used to send the Up/Open Command.

### 5.6.2 Prototype
```
ZStatus_t zclClosures_SendUpOpen( uint8 srcEP,
    afAddrType_t *dstAddr,
    uint8 disableDefaultRsp,
    uint8 seqNum );
```

### 5.6.3 Parameter Details
`srcEP` – The source endpoint.
`destAddr` – The destination address.
`disableDefaultRsp` - Disable Default Response command.
`seqNum` - The identification number for the transaction.

### 5.6.4 Return
`ZStatus_t` – enum found in ZComDef.h.

## 5.7 Down / Close Command (Window Covering)

### 5.7.1 Description
This function is used to send the Down/Close Command.

### 5.7.2 Prototype
```
ZStatus_t zclClosures_SendDownClose( uint8 srcEP,
 afAddrType_t *dstAddr,
 uint8 disableDefaultRsp,
 uint8 seqNum );
```

### 5.7.3 Parameter Details
`srcEP` – The source endpoint.
`destAddr` – The destination address.
`disableDefaultRsp` - Disable Default Response command.
`seqNum` - The identification number for the transaction.

### 5.7.4 Return
`ZStatus_t` – enum found in ZComDef.h.

 

## 5.8 Stop Command (Window covering)

### 5.8.1 Description
This function is used to send the Stop Command.

### 5.8.2 Prototype
```
ZStatus_t zclClosures_SendStop( uint8 srcEP,
  afAddrType_t *dstAddr,
  uint8 disableDefaultRsp,
        uint8 seqNum );
```

### 5.8.3 Parameter Details
`srcEP` – The source endpoint.
`destAddr` – The destination address.
`disableDefaultRsp` - Disable Default Response command.
`seqNum` - The identification number for the transaction.

### 5.8.4 Return
`ZStatus_t` – enum found in ZComDef.h.

## 5.9 Go to Lift Setpoint Command (Window Covering)

### 5.9.1 Description
This function is used to send the Go to Lift Setpoint Command.

### 5.9.2 Prototype
```
ZStatus_t zclClosures_SendGoToLiftSetpoint( uint8 srcEP,
  afAddrType_t *dstAddr,
  uint8 disableDefaultRsp,
  uint8 seqNum,
  uint8 liftSetpoint );
```

### 5.9.3 Parameter Details
`srcEP` – The source endpoint.
`destAddr` – The destination address.
`disableDefaultRsp` - Disable Default Response command.
`seqNum` - The identification number for the transaction.
`liftSetpoint` – Index of Lift Setpoint.

### 5.9.4 Return
`ZStatus_t` – enum found in ZComDef.h.

 

### 5.10 Go To Lift Value Command (Window Covering)

#### 5.10.1 Description
This function is used to send the Go to Lift Setpoint Command.

#### 5.10.2 Prototype
```
ZStatus_t zclClosures_SendGoToLiftValue( uint8 srcEP,
     afAddrType_t *dstAddr,
     uint8 disableDefaultRsp,
     uint8 seqNum,
     uint16 liftValue );
```

#### 5.10.3 Parameter Details
`srcEP` – The source endpoint.
`destAddr` – The destination address.
`disableDefaultRsp` - Disable Default Response command.
`seqNum` - The identification number for the transaction.
`liftValue` – The Lift value.

#### 5.10.4 Return
`ZStatus_t` – enum found in ZComDef.h.

### 5.11 Go To Lift Percentage Command (Window Covering)

#### 5.11.1 Description
This function is used to send the Go to Lift Percentage Command.

#### 5.11.2 Prototype
```
ZStatus_t zclClosures_SendGoToLiftPercentage( uint8 srcEP,
    afAddrType_t *dstAddr,
    uint8 disableDefaultRsp,
    uint8 seqNum,
    uint8 percentageLiftValue );
```

#### 5.11.3 Parameter Details
`srcEP` – The source endpoint.
`destAddr` – The destination address.
`disableDefaultRsp` - Disable Default Response command.
`seqNum` - The identification number for the transaction.
`percentageLiftValue` – The Percentage Lift value.

#### 5.11.4 Return
`ZStatus_t` – enum found in ZComDef.h.

 

### 5.12 Go to Tilt Setpoint Command (Window Covering)

#### 5.12.1 Description
This function is used to send the Go to Tilt Setpoint Command.

#### 5.12.2 Prototype
```
ZStatus_t zclClosures_SendGoToTiltSetpoint( uint8 srcEP,
  afAddrType_t *dstAddr,
  uint8 disableDefaultRsp,
  uint8 seqNum,
  uint8 tiltSetpoint );
```

#### 5.12.3 Parameter Details
`srcEP` – The source endpoint.
`destAddr` – The destination address.
`disableDefaultRsp` - Disable Default Response command.
`seqNum` - The identification number for the transaction.
`tiltSetpoint` – Index of Tilt Setpoint.

#### 5.12.4 Return
`ZStatus_t` – enum found in ZComDef.h.

### 5.13 Go To Tilt Value Command (Window Covering)

#### 5.13.1 Description
This function is used to send the Go to Tilt Setpoint Command.

#### 5.13.2 Prototype
```
ZStatus_t zclClosures_SendGoToTiltValue( uint8 srcEP,
     afAddrType_t *dstAddr,
     uint8 disableDefaultRsp,
     uint8 seqNum,
     uint16 tiltValue );
```

#### 5.13.3 Parameter Details
`srcEP` – The source endpoint.
`destAddr` – The destination address.
`disableDefaultRsp` - Disable Default Response command.
`seqNum` - The identification number for the transaction.
`tiltValue` – The Lift value.

#### 5.13.4 Return
`ZStatus_t` – enum found in ZComDef.h.

 

### 5.14 Go To Tilt Percentage Command (Window Covering)

#### 5.14.1 Description
This function is used to send the Go to Tilt Percentage Command.

#### 5.14.2 Prototype
```
ZStatus_t zclClosures_SendGoToTiltPercentage( uint8 srcEP,
    afAddrType_t *dstAddr,
    uint8 disableDefaultRsp,
    uint8 seqNum,
    uint8 percentageTiltValue );
```

#### 5.14.3 Parameter Details
`srcEP` – The source endpoint.
`destAddr` – The destination address.
`disableDefaultRsp` - Disable Default Response command.
`seqNum` - The identification number for the transaction.
`percentageTiltValue` – The Percentage Lift value.

#### 5.14.4 Return
`ZStatus_t` – enum found in ZComDef.h.

### 5.15 Program Setpoint Command (Window Covering)

#### 5.15.1 Description
This function is used to send the Program Setpoint Command, in order to program specific Lift or Tilt setpoint with certain value (VERSION 1), or to program specific Lift and Tilt setpoints with the current Lift and Tilt values (VERSION 2).

#### 5.15.2 Prototype
```
ZStatus_t zclClosures_SendProgramSetpoint( uint8 srcEP,
      afAddrType_t *dstAddr,
      uint8 disableDefaultRsp,
      uint8 seqNum,
programSetpointPayload_t *programSetpoint );
```

#### 5.15.3 Parameter Details
`srcEP` – The source endpoint.
`destAddr` – The destination address.
`disableDefaultRsp` - Disable Default Response command.
`seqNum` - The identification number for the transaction.
`programSetpoint` – The Setpoint to be programmed, which has the following fields:
`version` - Version of the Program Setpoint command
`setpointIndex` - Index of the Setpoint
`setpointValue` - Value of the Setpoint (used in VERSION 1 only)
`setpointType` - Type of the Setpoint; it should be either lift or tilt (used in VERSION 1 only)

#### 5.15.4 Return
`ZStatus_t` – enum found in ZComDef.h.

 

## 5.16 Door Lock Callback

### 5.16.1 Description
This callback is called to process an incoming Door Lock/Unlock command.

### 5.16.2 Prototype
```
typedef void (*zclClosures_DoorLock_t) ( zclIncoming_t *pInMsg,
                                         zclDoorLock_t *pInCmd )
```

### 5.16.3 Parameter Details
`pInMsg` – Pointer to incoming message.
`pInCmd` – Pointer to PIN/RFID code.

### 5.16.4 Return
None.

## 5.17 Door Lock Response Callback

### 5.17.1 Description
This callback is called to process an incoming Door Lock/Unlock response

### 5.17.2 Prototype
```
typedef ZStatus_t (*zclClosures_DoorLockRsp_t) ( zclIncoming_t *pInMsg,
                                                 uint8 status );
```

### 5.17.3 Parameter Details
`pInMsg` – Pointer to incoming message.
`status` – status of the response.

### 5.17.4 Return
ZCL_STATUS_SUCCESS or ZCL_STATUS_FAILURE. The status byte only indicates if the message has received successfully.

## 5.18 Unlock With Timeout Callback

### 5.18.1 Description
This callback is called to process an incoming Unlock With Timeout command : This command causes the lock device to unlock the door with a timeout parameter. After the time in seconds specified in the timeout field, the lock device will relock itself automatically.

### 5.18.2 Prototype
```
typedef ZStatus_t (*zclClosures_DoorLockUnlockWithTimeout_t)
                ( zclIncoming_t *pInMsg, zclDoorLockUnlockTimeout_t *pCmd );
```

### 5.18.3 Parameter Details
`pInMsg` – Pointer to incoming message.
`pCmd` – Pointer to command payload.

### 5.18.4 Return
`ZStatus_t` – enum found in ZComDef.h.

 

### 5.19 Get Log Record Callback

#### 5.19.1 Description
This callback is called to process an incoming Get Log Record command.

#### 5.19.2 Prototype
```
typedef ZStatus_t (*zclClosures_DoorLockGetLogRecord_t)
                  ( zclIncoming_t *pInMsg, zclDoorLockGetLogRecord_t *pCmd );
```

#### 5.19.3 Parameter Details
`pInMsg` – Pointer to incoming message.
`pCmd` – Pointer to command payload.

#### 5.19.4 Return
`ZStatus_t` – enum found in ZComDef.h.

### 5.20 Set PIN Code Callback

#### 5.20.1 Description
This callback is called to process an incoming Set PIN Code command.

#### 5.20.2 Prototype
```
typedef ZStatus_t (*zclClosures_DoorLockSetPINCode_t);
                     ( zclIncoming_t *pInMsg, zclDoorLockSetPINCode_t *pCmd )
```

#### 5.20.3 Parameter Details
`pInMsg` – Pointer to incoming message.
`pCmd` – Pointer to command payload.

#### 5.20.4 Return
`ZStatus_t` – enum found in ZComDef.h.

### 5.21 Get PIN Code command Callback

#### 5.21.1 Description
This callback is called to process an incoming Get PIN Code command.

#### 5.21.2 Prototype
```
typedef ZStatus_t (*zclClosures_DoorLockGetPINCode_t)
                     ( zclIncoming_t *pInMsg, zclDoorLockUserID_t *pCmd );
```

#### 5.21.3 Parameter Details
`pInMsg` – Pointer to incoming message.
`pCmd` – Pointer to command payload.

#### 5.21.4 Return
`ZStatus_t` – enum found in ZComDef.h.

 

### 5.22 Clear PIN Code Callback

#### 5.22.1 Description
This callback is called to process an incoming Clear PIN Code command.

#### 5.22.2 Prototype
```
typedef ZStatus_t (*zclClosures_DoorLockClearPINCode_t)
                     ( zclIncoming_t *pInMsg, zclDoorLockUserID_t *pCmd );
```

#### 5.22.3 Parameter Details
`pInMsg` – Pointer to incoming message.
`pCmd` – Pointer to command payload.

#### 5.22.4 Return
`ZStatus_t` – enum found in ZComDef.h.

### 5.23 Clear All PIN Codes Callback

#### 5.23.1 Description
This callback is called to process an incoming Clear All PIN Codes command.

#### 5.23.2 Prototype
```
typedef ZStatus_t (*zclClosures_DoorLockClearAllPINCodes_t)
                                          ( zclIncoming_t *pInMsg );
```

#### 5.23.3 Parameter Details
`pInMsg` – Pointer to incoming message.

#### 5.23.4 Return
`ZStatus_t` – enum found in ZComDef.h.

### 5.24 Set User Status Callback

#### 5.24.1 Description
This callback is called to process an incoming Set User Status command.

#### 5.24.2 Prototype
```
typedef ZStatus_t (*zclClosures_DoorLockSetUserStatus_t)
                  ( zclIncoming_t *pInMsg, zclDoorLockSetUserStatus_t *pCmd );
```

#### 5.24.3 Parameter Details
`pInMsg` – Pointer to incoming message.
`pCmd` – Pointer to command payload.

#### 5.24.4 Return
`ZStatus_t` – enum found in ZComDef.h.

 

### 5.25 Get User Status Callback

#### 5.25.1 Description
This callback is called to process an incoming Get User Status command.

#### 5.25.2 Prototype
```
typedef ZStatus_t (*zclClosures_DoorLockGetUserStatus_t)
                ( zclIncoming_t *pInMsg, zclDoorLockUserID_t *pCmd );
```

#### 5.25.3 Parameter Details
`pInMsg` – Pointer to incoming message.
`pCmd` – Pointer to command payload.

#### 5.25.4 Return
`ZStatus_t` – enum found in ZComDef.h.

### 5.26 Set Week Day Schedule Callback

#### 5.26.1 Description
This callback is called to process an incoming Set Week Day Schedule command.

#### 5.26.2 Prototype
```
typedef ZStatus_t (*zclClosures_DoorLockSetWeekDaySchedule_t)
           ( zclIncoming_t *pInMsg, zclDoorLockSetWeekDaySchedule_t *pCmd );
```

#### 5.26.3 Parameter Details
`pInMsg` – Pointer to incoming message.
`pCmd` – Pointer to command payload.

#### 5.26.4 Return
`ZStatus_t` – enum found in ZComDef.h.

### 5.27 Get Week Day Schedule Callback

#### 5.27.1 Description
This callback is called to process an incoming Get Week Day Schedule command.

#### 5.27.2 Prototype
```
typedef ZStatus_t (*zclClosures_DoorLockGetWeekDaySchedule_t)
                      ( zclIncoming_t *pInMsg, zclDoorLockSchedule_t *pCmd );
```

#### 5.27.3 Parameter Details
`pInMsg` – Pointer to incoming message.
`pCmd` – Pointer to command payload.

#### 5.27.4 Return
`ZStatus_t` – enum found in ZComDef.h.

 

### 5.28 Clear Week Day Schedule Callback

#### 5.28.1 Description
This callback is called to process an incoming Clear Week Day Schedule command.

#### 5.28.2 Prototype
```
typedef ZStatus_t (*zclClosures_DoorLockClearWeekDaySchedule_t)
                    ( zclIncoming_t *pInMsg, zclDoorLockSchedule_t *pCmd );
```
#### 5.28.3 Parameter Details
`pInMsg` – Pointer to incoming message.
`pCmd` – Pointer to command payload.

#### 5.28.4 Return
`ZStatus_t` – enum found in ZComDef.h.

### 5.29 Set Year Day Schedule Callback

#### 5.29.1 Description
This callback is called to process an incoming Set Year Day Schedule command.

#### 5.29.2 Prototype
```
typedef ZStatus_t (*zclClosures_DoorLockSetYearDaySchedule_t)
                     ( zclIncoming_t *pInMsg, zclDoorLockSchedule_t *pCmd );
```
#### 5.29.3 Parameter Details
`pInMsg` – Pointer to incoming message.
`pCmd` – Pointer to command payload.

#### 5.29.4 Return
`ZStatus_t` – enum found in ZComDef.h.

### 5.30 Get Year Day Schedule Callback

#### 5.30.1 Description
This callback is called to process an incoming Get Year Day Schedule command.

#### 5.30.2 Prototype
```
typedef ZStatus_t (*zclClosures_DoorLockGetYearDaySchedule_t)
                     ( zclIncoming_t *pInMsg, zclDoorLockSchedule_t *pCmd );
```
#### 5.30.3 Parameter Details
`pInMsg` – Pointer to incoming message.
`pCmd` – Pointer to command payload.

#### 5.30.4 Return
`ZStatus_t` – enum found in ZComDef.h.

 

### 5.31 Clear Year Day Schedule Callback

#### 5.31.1 Description
This callback is called to process an incoming Clear Year Day Schedule command.

#### 5.31.2 Prototype
```
typedef ZStatus_t (*zclClosures_DoorLockClearYearDaySchedule_t)
                    ( zclIncoming_t *pInMsg, zclDoorLockSchedule_t *pCmd );
```

#### 5.31.3 Parameter Details
`pInMsg` – Pointer to incoming message.
`pCmd` – Pointer to command payload.

#### 5.31.4 Return
`ZStatus_t` – enum found in ZComDef.h.

### 5.32 Set Holiday Schedule Callback

#### 5.32.1 Description
This callback is called to process an incoming Set Holiday Schedule command.

#### 5.32.2 Prototype
```
typedef ZStatus_t (*zclClosures_DoorLockSetHolidaySchedule_t)
           ( zclIncoming_t *pInMsg, zclDoorLockSetHolidaySchedule_t *pCmd );
```

#### 5.32.3 Parameter Details
`pInMsg` – Pointer to incoming message.
`pCmd` – Pointer to command payload.

#### 5.32.4 Return
`ZStatus_t` – enum found in ZComDef.h.

### 5.33 Get Holiday Schedule Callback

#### 5.33.1 Description
This callback is called to process an incoming Get Holiday Schedule command.

#### 5.33.2 Prototype
```
typedef ZStatus_t (*zclClosures_DoorLockGetHolidaySchedule_t)
           ( zclIncoming_t *pInMsg, zclDoorLockHolidayScheduleID_t *pCmd );
```

#### 5.33.3 Parameter Details
`pInMsg` – Pointer to incoming message.
`pCmd` – Pointer to command payload.

#### 5.33.4 Return
`ZStatus_t` – enum found in ZComDef.h.

 

### 5.34 Clear Holiday Schedule Callback

#### 5.34.1 Description
This callback is called to process an incoming Clear Holiday Schedule command.

#### 5.34.2 Prototype
```
typedef ZStatus_t (*zclClosures_DoorLockClearHolidaySchedule_t)
           ( zclIncoming_t *pInMsg, zclDoorLockHolidayScheduleID_t *pCmd );
```

#### 5.34.3 Parameter Details
`pInMsg` – Pointer to incoming message.
`pCmd` – Pointer to command payload.

#### 5.34.4 Return
`ZStatus_t` – enum found in ZComDef.h.

### 5.35 Set User Type Callback

#### 5.35.1 Description
This callback is called to process an incoming Set User Type command.

#### 5.35.2 Prototype
```
typedef ZStatus_t (*zclClosures_DoorLockSetUserType_t)
                  ( zclIncoming_t *pInMsg, zclDoorLockSetUserType_t *pCmd );
```

#### 5.35.3 Parameter Details
`pInMsg` – Pointer to incoming message.
`pCmd` – Pointer to command payload.

#### 5.35.4 Return
`ZStatus_t` – enum found in ZComDef.h.

### 5.36 Get User Type Callback

#### 5.36.1 Description
This callback is called to process an incoming Get User Type command.

#### 5.36.2 Prototype
```
typedef ZStatus_t (*zclClosures_DoorLockGetUserType_t)
                     ( zclIncoming_t *pInMsg, zclDoorLockUserID_t *pCmd );
```

#### 5.36.3 Parameter Details
`pInMsg` – Pointer to incoming message.
`pCmd` – Pointer to command payload.

#### 5.36.4 Return
`ZStatus_t` – enum found in ZComDef.h.

 

### 5.37 Set RFID Code Callback

#### 5.37.1 Description
This callback is called to process an incoming Set RFID Code command.

#### 5.37.2 Prototype
```
typedef ZStatus_t (*zclClosures_DoorLockSetRFIDCode_t)
                  ( zclIncoming_t *pInMsg, zclDoorLockSetRFIDCode_t *pCmd );
```

#### 5.37.3 Parameter Details
`pInMsg` – Pointer to incoming message.
`pCmd` – Pointer to command payload.

#### 5.37.4 Return
`ZStatus_t` – enum found in ZComDef.h.

### 5.38 Get RFID Code Callback

#### 5.38.1 Description
This callback is called to process an incoming Get RFID Code command.

#### 5.38.2 Prototype
```
typedef ZStatus_t (*zclClosures_DoorLockGetRFIDCode_t)
                    ( zclIncoming_t *pInMsg, zclDoorLockUserID_t *pCmd );
```

#### 5.38.3 Parameter Details
`pInMsg` – Pointer to incoming message.
`pCmd` – Pointer to command payload.

#### 5.38.4 Return
`ZStatus_t` – enum found in ZComDef.h.

### 5.39 Clear RFID Code Callback

#### 5.39.1 Description
This callback is called to process an incoming Clear RFID Code command.

#### 5.39.2 Prototype
```
typedef ZStatus_t (*zclClosures_DoorLockClearRFIDCode_t)
                        ( zclIncoming_t *pInMsg, zclDoorLockUserID_t *pCmd );
```

#### 5.39.3 Parameter Details
`pInMsg` – Pointer to incoming message.
`pCmd` – Pointer to command payload.

#### 5.39.4 Return
`ZStatus_t` – enum found in ZComDef.h.

 

### 5.40 Clear All RFID Codes Callback

#### 5.40.1 Description
This callback is called to process an incoming Clear All RFID Codes command.

#### 5.40.2 Prototype
```
typedef ZStatus_t (*zclClosures_DoorLockClearAllRFIDCodes_t)
                                                    ( zclIncoming_t *pInMsg );
```

#### 5.40.3 Parameter Details
`pInMsg` – Pointer to incoming message.

#### 5.40.4 Return
`ZStatus_t` – enum found in ZComDef.h.

### 5.41 Lock Door Response Callback

#### 5.41.1 Description
This callback is called to process an incoming Lock Door Response command.

#### 5.41.2 Prototype
```
typedef ZStatus_t (*zclClosures_DoorLockRsp_t)
                                    ( zclIncoming_t *pInMsg, uint8 status );
```

#### 5.41.3 Parameter Details
`pInMsg` – Pointer to incoming message.
`status` – Operation status SUCCESS or FAILURE.

#### 5.41.4 Return
`ZStatus_t` – enum found in ZComDef.h.

### 5.42 Unlock With Timeout Response Callback

#### 5.42.1 Description
This callback is called to process an incoming Unlock With Timeout Response command.

#### 5.42.2 Prototype
```
typedef ZStatus_t (*zclClosures_DoorLockUnlockWithTimeoutRsp_t)
                             ( zclIncoming_t *pInMsg, uint8 status );
```

#### 5.42.3 Parameter Details
`pInMsg` – Pointer to incoming message.
`status` – Operation status SUCCESS or FAILURE.

#### 5.42.4 Return
`ZStatus_t` – enum found in ZComDef.h.

 

### 5.43 Get Log Record Response Callback

#### 5.43.1 Description
This callback is called to process an incoming Get Log Record Response command.

#### 5.43.2 Prototype
```
typedef ZStatus_t (*zclClosures_DoorLockGetLogRecordRsp_t)
                ( zclIncoming_t *pInMsg, zclDoorLockGetLogRecordRsp_t *pCmd );
```

#### 5.43.3 Parameter Details
`pInMsg` – Pointer to incoming message.
`pCmd` – Pointer to command payload.

#### 5.43.4 Return
`ZStatus_t` – enum found in ZComDef.h.

### 5.44 Set PIN Code Response Callback

#### 5.44.1 Description
This callback is called to process an incoming Set PIN Code Response command.

#### 5.44.2 Prototype
```
typedef ZStatus_t (*zclClosures_DoorLockSetPINCodeRsp_t)
                                    ( zclIncoming_t *pInMsg, uint8 status );
```

#### 5.44.3 Parameter Details
`pInMsg` – Pointer to incoming message.
`status` – Operation status SUCCESS or FAILURE.

#### 5.44.4 Return
`ZStatus_t` – enum found in ZComDef.h.

### 5.45 Get PIN Code Response Callback

#### 5.45.1 Description
This callback is called to process an incoming Get PIN Code Response command.

#### 5.45.2 Prototype
```
typedef ZStatus_t (*zclClosures_DoorLockGetPINCodeRsp_t)
                  ( zclIncoming_t *pInMsg, zclDoorLockGetPINCodeRsp_t *pCmd );
```

#### 5.45.3 Parameter Details
`pInMsg` – Pointer to incoming message.
`pCmd` – Pointer to command payload.

#### 5.45.4 Return
`ZStatus_t` – enum found in ZComDef.h.

 

### 5.46 Clear PIN Code Response Callback

#### 5.46.1 Description
This callback is called to process an incoming Clear PIN Code Response command.

#### 5.46.2 Prototype
```
typedef ZStatus_t (*zclClosures_DoorLockClearPINCodeRsp_t)
                                    ( zclIncoming_t *pInMsg, uint8 status );
```

#### 5.46.3 Parameter Details
`pInMsg` – Pointer to incoming message.
`status` – Operation status SUCCESS or FAILURE.

#### 5.46.4 Return
`ZStatus_t` – enum found in ZComDef.h.

### 5.47 Clear All PIN Codes Response Callback

#### 5.47.1 Description
This callback is called to process an incoming Clear All PIN Codes Response command.

#### 5.47.2 Prototype
```
typedef ZStatus_t (*zclClosures_DoorLockClearAllPINCodesRsp_t)
                                ( zclIncoming_t *pInMsg, uint8 status );
```

#### 5.47.3 Parameter Details
`pInMsg` – Pointer to incoming message.
`status` – Operation status SUCCESS or FAILURE.

#### 5.47.4 Return
`ZStatus_t` – enum found in ZComDef.h.

### 5.48 Set User Status Response Callback

#### 5.48.1 Description
This callback is called to process an incoming Set User Status Response command.

#### 5.48.2 Prototype
```
typedef ZStatus_t (*zclClosures_DoorLockSetUserStatusRsp_t)
                                  ( zclIncoming_t *pInMsg, uint8 status );
```

#### 5.48.3 Parameter Details
`pInMsg` – Pointer to incoming message.
`status` – Operation status SUCCESS or FAILURE.

#### 5.48.4 Return
`ZStatus_t` – enum found in ZComDef.h.

 

### 5.49 Get User Status Response Callback

#### 5.49.1 Description
This callback is called to process an incoming Get User Status Response command.

#### 5.49.2 Prototype
```
typedef (*zclClosures_DoorLockGetUserStatusRsp_t)
             ( zclIncoming_t *pInMsg, zclDoorLockGetUserStatusRsp_t *pCmd );
```

#### 5.49.3 Parameter Details
`pInMsg` – Pointer to incoming message.
`pCmd` – Pointer to command payload.

#### 5.49.4 Return
`ZStatus_t` – enum found in ZComDef.h.

### 5.50 Set Week Day Schedule Response Callback

#### 5.50.1 Description
This callback is called to process an incoming Set Week Day Schedule Response command.

#### 5.50.2 Prototype
```
typedef ZStatus_t (*zclClosures_DoorLockSetWeekDayScheduleRsp_t)
                                    ( zclIncoming_t *pInMsg, uint8 status );
```

#### 5.50.3 Parameter Details
`pInMsg` – Pointer to incoming message.
`status` – Operation status SUCCESS or FAILURE.

#### 5.50.4 Return
`ZStatus_t` – enum found in ZComDef.h.

### 5.51 Get Week Day Schedule Response Callback

#### 5.51.1 Description
This callback is called to process an incoming Get Week Day Schedule Response command.

#### 5.51.2 Prototype
```
typedef ZStatus_t (*zclClosures_DoorLockGetWeekDayScheduleRsp_t)
        ( zclIncoming_t *pInMsg, zclDoorLockGetWeekDayScheduleRsp_t *pCmd );
```

#### 5.51.3 Parameter Details
`pInMsg` – Pointer to incoming message.
`pCmd` – Pointer to command payload.

#### 5.51.4 Return
`ZStatus_t` – enum found in ZComDef.h.

 

### 5.52 Clear Week Day Schedule Response Callback

#### 5.52.1 Description
This callback is called to process an incoming Clear Week Day Schedule Response command.

#### 5.52.2 Prototype
```
typedef ZStatus_t (*zclClosures_DoorLockClearWeekDayScheduleRsp_t)
                                    ( zclIncoming_t *pInMsg, uint8 status );
```
#### 5.52.3 Parameter Details
`pInMsg` – Pointer to incoming message.
`status` – Operation status SUCCESS or FAILURE.

#### 5.52.4 Return
`ZStatus_t` – enum found in ZComDef.h.

### 5.53 Clear Week Day Schedule Response Callback

#### 5.53.1 Description
This callback is called to process an incoming Clear Week Day Schedule Response command.

#### 5.53.2 Prototype
```
typedef ZStatus_t (*zclClosures_DoorLockClearWeekDayScheduleRsp_t)
                                    ( zclIncoming_t *pInMsg, uint8 status );
```
#### 5.53.3 Parameter Details
`pInMsg` – Pointer to incoming message.
`status` – Operation status SUCCESS or FAILURE.

#### 5.53.4 Return
`ZStatus_t` – enum found in ZComDef.h.

### 5.54 Set Year Day Schedule Response Callback

#### 5.54.1 Description
This callback is called to process an incoming Set Year Day Schedule Response command.

#### 5.54.2 Prototype
```
typedef ZStatus_t (*zclClosures_DoorLockSetYearDayScheduleRsp_t)
                                    ( zclIncoming_t *pInMsg, uint8 status );
```
#### 5.54.3 Parameter Details
`pInMsg` – Pointer to incoming message.
`status` – Operation status SUCCESS or FAILURE.

#### 5.54.4 Return
`ZStatus_t` – enum found in ZComDef.h.

 

### 5.55 Get Year Day Schedule Response Callback

#### 5.55.1 Description
This callback is called to process an incoming Get Year Day Schedule Response command.

#### 5.55.2 Prototype
```
typedef ZStatus_t (*zclClosures_DoorLockGetYearDayScheduleRsp_t)
        ( zclIncoming_t *pInMsg, zclDoorLockGetYearDayScheduleRsp_t *pCmd );
```

#### 5.55.3 Parameter Details
`pInMsg` – Pointer to incoming message.
`pCmd` – Pointer to command payload.

#### 5.55.4 Return
`ZStatus_t` – enum found in ZComDef.h.

### 5.56 Clear Year Day Schedule Response Callback

#### 5.56.1 Description
This callback is called to process an incoming Clear Year Day Schedule Response command.

#### 5.56.2 Prototype
```
typedef ZStatus_t (*zclClosures_DoorLockClearYearDayScheduleRsp_t)
                                    ( zclIncoming_t *pInMsg, uint8 status );
```

#### 5.56.3 Parameter Details
`pInMsg` – Pointer to incoming message.
`status` – Operation status SUCCESS or FAILURE.

#### 5.56.4 Return
`ZStatus_t` – enum found in ZComDef.h.

### 5.57 Set Holiday Schedule Response Callback

#### 5.57.1 Description
This callback is called to process an incoming Set Holiday Schedule Response command.

#### 5.57.2 Prototype
```
typedef ZStatus_t (*zclClosures_DoorLockSetHolidayScheduleRsp_t)
                                   ( zclIncoming_t *pInMsg, uint8 status );
```

#### 5.57.3 Parameter Details
`pInMsg` – Pointer to incoming message.
`status` – Operation status SUCCESS or FAILURE.

#### 5.57.4 Return
`ZStatus_t` – enum found in ZComDef.h.

 

### 5.58 Get Holiday Schedule Response Callback

#### 5.58.1 Description
This callback is called to process an incoming Get Holiday Schedule Response command.

#### 5.58.2 Prototype
```
typedef ZStatus_t (*zclClosures_DoorLockGetHolidayScheduleRsp_t)
         ( zclIncoming_t *pInMsg, zclDoorLockGetHolidayScheduleRsp_t *pCmd );
```

#### 5.58.3 Parameter Details
`pInMsg` – Pointer to incoming message.
`pCmd` – Pointer to command payload.

#### 5.58.4 Return
`ZStatus_t` – enum found in ZComDef.h.

### 5.59 Clear Holiday Schedule Response Callback

#### 5.59.1 Description
This callback is called to process an incoming Clear Holiday Schedule Response command.

#### 5.59.2 Prototype
```
typedef ZStatus_t (*zclClosures_DoorLockClearHolidayScheduleRsp_t)
                                  ( zclIncoming_t *pInMsg, uint8 status );
```

#### 5.59.3 Parameter Details
`pInMsg` – Pointer to incoming message.
`status` – Operation status SUCCESS or FAILURE.

#### 5.59.4 Return
`ZStatus_t` – enum found in ZComDef.h.

### 5.60 Set User Type Response Callback

#### 5.60.1 Description
This callback is called to process an incoming Set User Type Response command.

#### 5.60.2 Prototype
```
typedef ZStatus_t (*zclClosures_DoorLockSetUserTypeRsp_t)
                                    ( zclIncoming_t *pInMsg, uint8 status );
```

#### 5.60.3 Parameter Details
`pInMsg` – Pointer to incoming message.
`status` – Operation status SUCCESS or FAILURE.

#### 5.60.4 Return
`ZStatus_t` – enum found in ZComDef.h.

 

### 5.61 Get User Type Response Callback

#### 5.61.1 Description
This callback is called to process an incoming Get User Type Response command.

#### 5.61.2 Prototype
```
typedef ZStatus_t (*zclClosures_DoorLockGetUserTypeRsp_t)
                ( zclIncoming_t *pInMsg, zclDoorLockGetUserTypeRsp_t *pCmd );
```

#### 5.61.3 Parameter Details
`pInMsg` – Pointer to incoming message.
`pCmd` – Pointer to command payload.

#### 5.61.4 Return
`ZStatus_t` – enum found in ZComDef.h.

### 5.62 Set RFID Code Response Callback

#### 5.62.1 Description
This callback is called to process an incoming Set RFID Code Response command.

#### 5.62.2 Prototype
```
typedef ZStatus_t (*zclClosures_DoorLockSetRFIDCodeRsp_t)
                                    ( zclIncoming_t *pInMsg, uint8 status );
```

#### 5.62.3 Parameter Details
`pInMsg` – Pointer to incoming message.
`status` – Operation status SUCCESS or FAILURE.

#### 5.62.4 Return
`ZStatus_t` – enum found in ZComDef.h.

### 5.63 Get RFID Code Response Callback

#### 5.63.1 Description
This callback is called to process an incoming Get RFID Code Response command.

#### 5.63.2 Prototype
```
typedef ZStatus_t (*zclClosures_DoorLockGetRFIDCodeRsp_t)
                ( zclIncoming_t *pInMsg, zclDoorLockGetRFIDCodeRsp_t *pCmd );
```

#### 5.63.3 Parameter Details
`pInMsg` – Pointer to incoming message.
`pCmd` – Pointer to command payload.

#### 5.63.4 Return
`ZStatus_t` – enum found in ZComDef.h.

 

### 5.64 Clear RFID Code Response Callback

#### 5.64.1 Description
This callback is called to process an incoming Clear RFID Code Response command.

#### 5.64.2 Prototype
```
typedef ZStatus_t (*zclClosures_DoorLockClearRFIDCodeRsp_t)
                                    ( zclIncoming_t *pInMsg, uint8 status );
```

#### 5.64.3 Parameter Details
`pInMsg` – Pointer to incoming message.
`status` – Operation status SUCCESS or FAILURE.

#### 5.64.4 Return
`ZStatus_t` – enum found in ZComDef.h.

### 5.65 Clear All RFID Codes Callback

#### 5.65.1 Description
This callback is called to process an incoming Clear All RFID Codes command.

#### 5.65.2 Prototype
```
typedef ZStatus_t (*zclClosures_DoorLockClearAllRFIDCodesRsp_t)
                                     ( zclIncoming_t *pInMsg, uint8 status );
```

#### 5.65.3 Parameter Details
`pInMsg` – Pointer to incoming message.
`status` – Operation status SUCCESS or FAILURE.

#### 5.65.4 Return
`ZStatus_t` – enum found in ZComDef.h.

### 5.66 Operation Event Notification Callback

#### 5.66.1 Description
This callback is called to process an incoming Operation Event Notification command.

#### 5.66.2 Prototype
```
typedef ZStatus_t (*zclClosures_DoorLockOperationEventNotification_t)
    ( zclIncoming_t *pInMsg, zclDoorLockOperationEventNotification_t *pCmd );
```

#### 5.66.3 Parameter Details
`pInMsg` – Pointer to incoming message.
`pCmd` – Pointer to command payload.

#### 5.66.4 Return
`ZStatus_t` – enum found in ZComDef.h.

 

### 5.67 ProgrammingEvent Notification Callback

#### 5.67.1 Description
This callback is called to process an incoming Programming Event Notification command.

#### 5.67.2 Prototype
```
typedef ZStatus_t (*zclClosures_DoorLockProgrammingEventNotification_t)
  ( zclIncoming_t *pInMsg, zclDoorLockProgrammingEventNotification_t *pCmd );
```

#### 5.67.3 Parameter Details
`pInMsg` – Pointer to incoming message.
`pCmd` – Pointer to command payload.

#### 5.67.4 Return
`ZStatus_t` – enum found in ZComDef.h.

### 5.68 Window Covering cluster basic Callback

#### 5.68.1 Description
This callback is called to process an incoming Window Covering cluster basic command.

#### 5.68.2 Prototype
```
typedef void (*zclClosures_WindowCoveringSimple_t) ( void );
```

#### 5.68.3 Parameter Details
`None.`

#### 5.68.4 Return
`None.`

### 5.69 Window Covering cluster goto percentage Callback

#### 5.69.1 Description
This callback is called to process an incoming Window Covering cluster goto percentage command.

#### 5.69.2 Prototype
```
typedef bool (*zclClosures_WindowCoveringGotoPercentage_t)
                                                    ( uint8 percentage );
```

#### 5.69.3 Parameter Details
`Percentage` – Percentage value.

#### 5.69.4 Return
`bool` – TRUE or FALSE.

 

### 5.70 Window Covering cluster goto value Callback

#### 5.70.1 Description
This callback is called to process an incoming Window Covering cluster goto value command.

#### 5.70.2 Prototype
```
typedef bool (*zclClosures_WindowCoveringGotoValue_t) ( uint16 value );
```

#### 5.70.3 Parameter Details
`value` – Desired value..

#### 5.70.4 Return
`bool` – TRUE or FALSE.

### 5.71 Window Covering cluster goto setpoint Callback

#### 5.71.1 Description
This callback is called to process an incoming Window Covering cluster goto setpoint command.

#### 5.71.2 Prototype
```
typedef uint8 (*zclClosures_WindowCoveringGotoSetpoint_t) ( uint8 index );
```

#### 5.71.3 Parameter Details
`index` – index value.

#### 5.71.4 Return
`uint8` – 0 to 255 value.

### 5.72 Window Covering cluster program setpoint Callback

#### 5.72.1 Description
This callback is called to process an incoming Window Covering cluster program setpoint command.

#### 5.72.2 Prototype
```
typedef bool (*zclClosures_WindowCoveringProgramSetpoint_t)
                                    ( programSetpointPayload_t *setpoint );
```

#### 5.72.3 Parameter Details
`setpoint` – Pointer to command payload.

#### 5.72.4 Return
`bool` – TRUE or FALSE.

 

# 6. Protocol Interfaces Functional Domain

## 6.1 Introduction

The Protocol Interfaces functional domain provides the following two clusters:

- Generic Tunnel cluster
- BACnet Protocol Tunnel cluster
- 11073 Protocol Tunnel cluster

The Generic Tunnel cluster is used when an associated protocol specific tunnel wishes to find out the ZigBee address of the Generic Tunnel server cluster representing a protocol-specific device with a given protocol address. The BACnet Protocol Tunnel cluster is used when a BACnet network layer wishes to transfer a BACnet NPDU across a ZigBee tunnel to another BACnet network layer. The 11073 Protocol Tunnel cluster is used when a 11073 network layer wishes to transfer an 11073 APDU and associated metadata across a ZigBee tunnel to another 11073 network layer.

The Generic Tunnel cluster provides the minimum common commands and attributes required to tunnel any protocol. The supported commands are:

- Match Protocol Address
- Match Protocol Address Response
- Advertise Protocol Address

The BACnet Protocol Tunnel cluster provides the commands and attributes to tunnel the BACnet protocol. The only supported command is:

- Transfer NPDU

The 11073 Protocol Tunnel cluster provides the commands and attributes to tunnel the 11073 protocol. The supported commands are:

- Transfer APDU
- Connect Request
- Disconnect Request
- Connect Status Notification

The BACnet Protocol Tunnel cluster requires the Generic Tunnel cluster's *MaximumIncomingTransferSize* attribute and *MaximumOutgoingTransferSize* attribute to be equal to or greater than 504 octets, hence, the *MAX_TRANSFER_SIZE* (defined in ZDConfig.h header file) should be set accordingly and the Fragmentation feature should be enabled by including the *ZIGBEE_FRAGMENTATION* compile flag in the project file. Similarly, the 11073 Protocol Tunnel cluster requires these two attributes to be equal to or greater than the maximum APDU size specified in the applicable ISO/IEEE 11073 device specialization document.

The Protocol Interfaces functional domain is implemented by zcl_pi.c and zcl_pi.h files.

 

## 6.2 Send Match Protocol Address Command (Generic Tunnel)

### 6.2.1 Description
This function is used to send a Match Protocol Address command. This command is used when an associated protocol specific tunnel wishes to find out the ZigBee address of the Generic Tunnel server cluster representing a protocol-specific device with a given protocol address. The command is typically multicast to a group of inter-communicating Generic Tunnel clusters.

### 6.2.2 Prototype

```
ZStatus_t zclPI_Send_MatchProtocolAddrCmd( uint8 srcEP,
                                           afAddrType_t *dstAddr,
                                           uint8 len,
                                           uint8 *protocolAddr,
                                           uint8 disableDefaultRsp,
                                           uint8 seqNum );
```

### 6.2.3 Parameter Details
`srcEP` – The source endpoint.
`destAddr` – The destination address.
`len` – The length of the Protocol Address.
`protocolAddr` – The Protocol Address.
`disableDefaultRsp` - Disable Default Response command.
`seqNum` - The identification number for the transaction.

### 6.2.4 Return
`ZStatus_t` – enum found in ZComDef.h.

## 6.3 Send Match Protocol Address Response (Generic Tunnel)

### 6.3.1 Description
This function is used to send a Match Protocol Address Response. This response is sent back upon receipt of a Match Protocol Address command to indicate that the Protocol Address was successfully matched.

### 6.3.2 Prototype

```
ZStatus_t zclPI_Send_MatchProtocolAddrRsp( uint8 srcEP,
                                           afAddrType_t *dstAddr,
                                           uint8 *ieeeAddr,
                                           uint8 len,
                                           uint8 *protocolAddr,
                                           uint8 disableDefaultRsp,
                                           uint8 seqNum );
```

### 6.3.3 Parameter Details
`srcEP` – The source endpoint.
`destAddr` – The destination address.
`ieeeAddr` – The Device Address.
`len` – The length of the Protocol Address.
`protocolAddr` – The Protocol Address.
`disableDefaultRsp` - Disable Default Response command.
`seqNum` - The identification number for the transaction.

### 6.3.4 Return

 

`ZStatus_t` – enum found in ZComDef.h.

### 6.4 Send Advertise Protocol Address Command (Generic Tunnel)

#### 6.4.1 Description
This function is used to send an Advertise Protocol Address Command. This command is sent out typically upon startup or whenever the Protocol Address attribute changes. It is typically multicast to a group of inter-communicating Generic Tunnel clusters.

#### 6.4.2 Prototype

```
ZStatus_t zclPI_Send_AdvertiseProtocolAddrCmd( uint8 srcEP,
                                               afAddrType_t *dstAddr,
                                               uint8 *ieeeAddr,
                                               uint8 len,
                                               uint8 *protocolAddr,
                                               uint8 disableDefaultRsp,
                                               uint8 seqNum );
```

#### 6.4.3 Parameter Details
`srcEP` – The source endpoint.
`destAddr` – The destination address.
`ieeeAddr` – The Device Address.
`len` – The length of the Protocol Address.
`protocolAddr` – The Protocol Address.
`disableDefaultRsp` - Disable Default Response command.
`seqNum` - The identification number for the transaction.

#### 6.4.4 Return
`ZStatus_t` – enum found in ZComDef.h.

### 6.5 Send BACnet Transfer NPDU Command (BACnet Protocol Tunnel)

#### 6.5.1 Description
This function is used to send a BACnet Transfer NPDU Command. This command is used when a BACnet network layer wishes to transfer a BACnet NPDU across a ZigBee tunnel to another BACnet network layer.

#### 6.5.2 Prototype

```
ZStatus_t zclPI_Send_BACnetTransferNPDUCmd( uint8 srcEP,
                                            afAddrType_t *dstAddr,
                                            uint8 len,
                                            uint8 *npdu,
                                            uint8 disableDefaultRsp,
                                            uint8 seqNum );
```

#### 6.5.3 Parameter Details
`srcEP` – The source endpoint.
`destAddr` – The destination address.
`len` – The length of the NPDU.
`npdu` – The NPDU to be transferred.
`disableDefaultRsp` - Disable Default Response command.
`seqNum` - The identification number for the transaction.

 

#### 6.5.4 Return
`ZStatus_t` – enum found in ZComDef.h.

### 6.6 Send 11073 Transfer APDU Command (11073 Protocol Tunnel)

#### 6.6.1 Description
This function is used to send an 11073 Transfer APDU Command. This command is used when an 11073 network layer wishes to transfer an 11073 APDU across a ZigBee tunnel to another 11073 network layer.

#### 6.6.2 Prototype

```
ZStatus_t zclPI_Send_11073TransferAPDUCmd( uint8 srcEP,
                                           afAddrType_t *dstAddr,
                                           uint8 len,
                                           uint8 *apdu,
                                           uint8 seqNum );
```

#### 6.6.3 Parameter Details
`srcEP` – The source endpoint.
`destAddr` - The destination address.
`len` - The length of the APDU.
`apdu` - The APDU to be transferred.
`seqNum` - The identification number for the transaction.

#### 6.6.4 Return
`ZStatus_t` – enum found in ZComDef.h.

### 6.7 Send 11073 Connect Request Command (11073 Protocol Tunnel)

#### 6.7.1 Description
This function is used to send an 11073 Connect Request Command. This command is generated when a Data Management device wishes to connect to an 11073 agent device. This may be in response to receiving a connect status notification command from that agent device with the connect status field set to RECONNECT_REQUEST.

#### 6.7.2 Prototype

```
ZStatus_t zclPI_Send_11073ConnectReq( uint8 srcEP,
                                      afAddrType_t *dstAddr,
                                      uint8 connectCtrl,
                                      uint16 idleTimeout,
                                      uint8 *managerAddr,
                                      uint8 managerEP,
                                      uint8 disableDefaultRsp,
                                      uint8 seqNum );
```

#### 6.7.3 Parameter Details
`srcEP` – The source endpoint.
`destAddr` - The destination address.
`connectCtrl` - The connect control.
`idleTimeout` - The inactivity time (in minutes) which Data Management device will wait w/o receiving any data before it disconnects.
`managerAddr`- The IEEE address (64-bit) of Data Management device transmitting this frame.

 

managerEP – The source endpoint from which Data Management device is transmitting this frame.
disableDefaultRsp - Disable Default Response command.
seqNum - The identification number for the transaction.

#### 6.7.4 Return
ZStatus_t – enum found in ZComDef.h.

### 6.8 Send 11073 Disconnect Request Command (11073 Protocol Tunnel)

#### 6.8.1 Description
This function is used to send an 11073 Disconnect Request Command. This command is generated when a Data Management device wishes to disconnect a tunnel connection existing on an agent device.

#### 6.8.2 Prototype

```
ZStatus_t zclPI_Send_11073DisconnectReq( uint8 srcEP,
                                         afAddrType_t *dstAddr,
                                         uint8 *managerAddr,
                                         uint8 disableDefaultRsp,
                                         uint8 seqNum );
```

#### 6.8.3 Parameter Details
srcEP – The source endpoint.
destAddr – The destination address.
managerAddr – The IEEE address (64-bit) of Data Management device transmitting this frame.
disableDefaultRsp - Disable Default Response command.
seqNum - The identification number for the transaction.

#### 6.8.4 Return
ZStatus_t – enum found in ZComDef.h.

### 6.9 Send 11073 Connect Status Notification Command (11073 Protocol Tunnel)

#### 6.9.1 Description
This function is used to send an 11073 Connect Status Notification Command. This command is generated by an agent device in response to a connect request command, disconnect command, or in response to some other event that causes the tunnel to become connected or disconnected. It is also sent by the agent device to request the Data Management device to reconnect a tunnel.

#### 6.9.2 Prototype

```
ZStatus_t zclPI_Send_11073ConnectStatusNoti( uint8 srcEP,
                                             afAddrType_t *dstAddr,
                                             uint8 connectStatus,
                                             uint8 disableDefaultRsp,
                                             uint8 seqNum );
```

#### 6.9.3 Parameter Details
srcEP – The source endpoint.
destAddr – The destination address.
connectStatus – The connect status.
disableDefaultRsp - Disable Default Response command.

 

seqNum - The identification number for the transaction.

#### 6.9.4 Return
ZStatus_t – enum found in ZComDef.h.

### 6.10 Register Application Command Callbacks

#### 6.10.1 Description
This function is used to register an Application's Command callbacks with the Protocol Interfaces functional domain.

#### 6.10.2 Prototype

```
ZStatus_t zclPI_RegisterCmdCallbacks( uint8 endpoint,
                                      zclPI_AppCallbacks_t *callbacks);
```

#### 6.10.3 Parameter Details
endpoint – The application's endpoint.
callbacks - Pointer to the callback records.

#### 6.10.4 Return
ZStatus_t – enum found in ZComDef.h.

### 6.11 Match Protocol Address Callback

#### 6.11.1 Description
This callback is called to process an incoming Match Protocol Address command.

#### 6.11.2 Prototype

```
typedef void (*zclPICB_MatchProtocolAddr_t)( zclPIMatchProtocolAddr_t*pCmd );
```

#### 6.11.3 Parameter Details
pCmd – received command, which has the following fields:

- srcAddr – The requestor's address.
- seqNum – The sequence number received with the command.
- len – The length of the Protocol Address.
- protocolAddr – The Protocol Address.

#### 6.11.4 Return
None.

### 6.12 Match Protocol Address Response Callback

#### 6.12.1 Description
This callback is called to process an incoming Match Protocol Address Response. . This means that this application has sent the corresponding command for this response.

#### 6.12.2 Prototype

```
typedef void (*zclPICB_MatchProtocolAddrRsp_t)(
                                        zclPIMatchProtocolAddrRsp_t*pRsp );
```

 

### 6.12.3 Parameter Details
pRsp – received response, which has the following fields:
srcAddr – The requestor's address.
ieeeAddr – The Device Address.
len – The length of the Protocol Address.
protocolAddr – The Protocol Address.

### 6.12.4 Return
None.

## 6.13 Advertise Protocol Address Callback

### 6.13.1 Description
This callback is called to process an incoming Advertise Protocol Address command.

### 6.13.2 Prototype

```
typedef void (*zclPICB_AdvertiseProtocolAddr_t)(
                                          zclPIAdvertiseProtocolAddr_t*pCmd );
```

### 6.13.3 Parameter Details
pCmd – received command, which has the following fields:
srcAddr – The requestor's address.
len – The length of the Protocol Address.
protocolAddr – The Protocol Address.

### 6.13.4 Return
None.

## 6.14 BACnet Transfer NPDU Callback

### 6.14.1 Description
This callback is called to process an incoming BACnet Transfer NPDU command.

### 6.14.2 Prototype

```
typedef void (*zclPICB_BACnetTransferNPDU_t)
                                      ( zclBACnetTransferNPDU_t *pCmd );
```

### 6.14.3 Parameter Details
pCmd – received command, which has the following fields:
srcAddr – The requestor's address.
len – The length of the BACnet NPDU.
npdu – The received BACnet NPDU.

### 6.14.4 Return
None.

 

### 6.15 11073 Transfer APDU Callback

#### 6.15.1 Description
This callback is called to process an incoming 11073 Transfer APDU command.

#### 6.15.2 Prototype

```
typedef void (*zclPICB_11073TransferAPDU_t)( zcl11073TransferAPDU_t *pCmd );
```

#### 6.15.3 Parameter Details
`pCmd` – received command, which has the following fields:
`srcAddr` – The requestor's address.
`len` – The length of the 11073 APDU.
`apdu` – The received 11073 APDU.

#### 6.15.4 Return
None.

### 6.16 11073 Connect Request Callback

#### 6.16.1 Description
This callback is called to process an incoming 11073 Connect Request command.

#### 6.16.2 Prototype

```
typedef void (*zclPICB_11073ConnectReq_t)( zcl11073ConnectReq_t *pCmd );
```

#### 6.16.3 Parameter Details
`pCmd` – received command, which has the following fields:
`srcAddr` – The requestor's address.
`seqNum` – The sequence number received with the command.
`connectCtrl` – The connect control.
`idleTimeout` – The inactivity time (in minutes) which Data Management device will wait w/o receiving any data before it disconnects.
`managerAddr` – The IEEE address (64-bit) of Data Management device transmitting this frame.
`managerEP`– The source endpoint from which Data Management device is transmitting this frame.

#### 6.16.4 Return
None.

### 6.17 11073 Disconnect Request Callback

#### 6.17.1 Description
This callback is called to process an incoming 11073 Disconnect Request command.

#### 6.17.2 Prototype

```
typedef void (*zclPICB_11073DisconnectReq_t)( zcl11073DisconnectReq_t *pCmd);
```

 

### 6.17.3 Parameter Details

`pCmd` – received command, which has the following fields:
`srcAddr` – The requestor's address.
`seqNum` – The sequence number received with the command.
`managerAddr` – The IEEE address (64-bit) of Data Management device transmitting this frame.

### 6.17.4 Return

None.

## 6.18 11073 Connect Status Notification Callback

### 6.18.1 Description

This callback is called to process an incoming 11073 Connect Status Notification command.

### 6.18.2 Prototype

```
typedef void (*zclPICB_11073ConnectStatusNoti_t)(
                                          zcl11073ConnectStatusNoti_t *pCmd );
```

### 6.18.3 Parameter Details

`pRsp` – received command, which has the following fields:
`srcAddr` – The requestor's address.
`connectStatus`– The connect status.

### 6.18.4 Return

None.

 

# 7. Touchlink Commissioning

## 7.1 Register Touchlink Inter-PAN command callbacks.

### 7.1.1 Description
This function is used to register inter-PAN command callbacks.

### 7.1.2 Prototype

```
ZStatus_t bdbTL_RegisterInterPANCmdCallbacks
                                (bdbTL_InterPANCallbacks_t *callbacks);
```

### 7.1.3 Parameter Details
`callbacks` – Pointer to inter-PAN callback record.

### 7.1.4 Return
`ZStatus_t` – enum found in ZComDef.h.

## 7.2 Send Scan Request

### 7.2.1 Description
This function is used to send a Scan Request.

### 7.2.2 Prototype

```
ZStatus_t bdbTL_Send_ScanReq (uint8 srcEP, afAddrType_t *dstAddr,
                                    bdbTLScanReq_t *pReq, uint8 seqNum);
```

### 7.2.3 Parameter Details
`srcEP` – The source endpoint.
`dstAddr` - The destination address.
`pReq`– Pointer to request parameters.
`seqNum` - The identification number for the transaction.

### 7.2.4 Return
`ZStatus_t` – enum found in ZComDef.h.

## 7.3 Device Information Request

### 7.3.1 Description
This function is used to send a Device Information Request.

### 7.3.2 Prototype

```
ZStatus_t bdbTL_Send_DeviceInfoReq (uint8 srcEP, afAddrType_t *dstAddr,
                                  bdbTLDeviceInfoReq_t *pReq, uint8 seqNum);
```

### 7.3.3 Parameter Details
`srcEP` – The source endpoint.
`dstAddr` - The destination address.
`pReq`– Pointer to request parameters.
`seqNum` - The identification number for the transaction.

 

#### 7.3.4 Return
`ZStatus_t` – enum found in ZComDef.h.

### 7.4 Send Identify Request

#### 7.4.1 Description
This function is used to send an Identify Request.

#### 7.4.2 Prototype

```
ZStatus_t bdbTL_Send_IndentifyReq (uint8 srcEP, afAddrType_t *dstAddr,
                                   bdbTLIdentifyReq_t *pReq, uint8 seqNum);
```

#### 7.4.3 Parameter Details
`srcEP` – The source endpoint.
`dstAddr` – The destination address.
`pReq`– Pointer to request parameters.
`seqNum` - The identification number for the transaction.

#### 7.4.4 Return
`ZStatus_t` – enum found in ZComDef.h.

### 7.5 Send Reset to Factory New Request

#### 7.5.1 Description
This function is used to send a Reset to Factory New Request.

#### 7.5.2 Prototype

```
ZStatus_t bdbTL_Send_ResetToFNReq (uint8 srcEP, afAddrType_t *dstAddr,
                                   bdbTLResetToFNReq_t *pReq, uint8 seqNum);
```

#### 7.5.3 Parameter Details
`srcEP` – The source endpoint.
`dstAddr` – The destination address.
`pReq`– Pointer to request parameters.
`seqNum` - The identification number for the transaction.

#### 7.5.4 Return
`ZStatus_t` – enum found in ZComDef.h.

### 7.6 Send a Network Start Request

#### 7.6.1 Description
This function is used to send a Network Start Request.

#### 7.6.2 Prototype

```
ZStatus_t bdbTL_Send_NwkStartReq (uint8 srcEP, afAddrType_t *dstAddr,
                                  bdbTLNwkStartReq_t *pRsp, uint8 seqNum);
```

 

#### 7.6.3 Parameter Details
srcEP – The source endpoint.
dstAddr – The destination address.
pReq– Pointer to request parameters.
seqNum - The identification number for the transaction.

#### 7.6.4 Return
ZStatus_t – enum found in ZComDef.h.

### 7.7 Send Network Join Router/End Device Request

#### 7.7.1 Description
This function is used to send a Network Join Router/End Device Request.

#### 7.7.2 Prototype

```
ZStatus_t bdbTL_Send_NwkJoinReq (uint8 srcEP, afAddrType_t *dstAddr,
                            bdbTLNwkJoinReq_t *pReq, uint8 cmd, uint8 seqNum);
```

#### 7.7.3 Parameter Details
srcEP – The source endpoint.
dstAddr – The destination address.
pReq – Pointer to request parameters.
cmd – Join router request or join end device request.
seqNum - The identification number for the transaction.

#### 7.7.4 Return
ZStatus_t – enum found in ZComDef.h.

### 7.8 Send Network Update Request

#### 7.8.1 Description
This function is used to send a Network Update Request.

#### 7.8.2 Prototype

```
ZStatus_t bdbTL_Send_NwkUpdateReq(uint8 srcEP, afAddrType_t *dstAddr,
                                  bdbTLNwkUpdateReq_t *pReq, uint8 seqNum);
```

#### 7.8.3 Parameter Details
srcEP – The source endpoint.
dstAddr – The destination address.
pReq– Pointer to request parameters.
seqNum - The identification number for the transaction.

#### 7.8.4 Return
ZStatus_t – enum found in ZComDef.h.

 

## 7.9 Send Get Group Identifiers Request

### 7.9.1 Description
This function is used to send a Get Group Identifiers Request.

### 7.9.2 Prototype

```
ZStatus_t bdbTL_Send_GetEPListReq (uint8 srcEP, afAddrType_t *dstAddr,
         bdbTLGetEPListReq_t *pReq, uint8 disableDefaultRsp, uint8 seqNum);
```

### 7.9.3 Parameter Details
`srcEP` – The source endpoint.
`dstAddr` - The destination address.
`pReq`– Pointer to response parameters.
`disableDefaultRsp` – whether to disable the Default Response command.
`seqNum` - The identification number for the transaction.

### 7.9.4 Return
`ZStatus_t` – enum found in ZComDef.h.

## 7.10 Send Get Endpoint List Request

### 7.10.1 Description
This function is used to send a Get Endpoint List Request.

### 7.10.2 Prototype

```
ZStatus_t bdbTL_Send_GetGrpIDsReq (uint8 srcEP, afAddrType_t *dstAddr,
         bdbTLGetGrpIDsReq_t *pReq, uint8 disableDefaultRsp, uint8 seqNum);
```

### 7.10.3 Parameter Details
`srcEP` – The source endpoint.
`dstAddr` - The destination address.
`pReq`– Pointer to request parameters.
`disableDefaultRsp` – whether to disable the Default Response command.
`seqNum` - The identification number for the transaction.

### 7.10.4 Return
`ZStatus_t` – enum found in ZComDef.h.

## 7.11 Send Scan Response

### 7.11.1 Description
This function is used to send a Scan Response.

### 7.11.2 Prototype

```
ZStatus_t bdbTL_Send_ScanRsp (uint8 srcEP, afAddrType_t *dstAddr,
                              bdbTLScanRsp_t *pRsp, uint8 seqNum);
```

 

#### 7.11.3 Parameter Details
srcEP – The source endpoint.
dstAddr – The destination address.
pRsp– Pointer to response parameters.
seqNum - The identification number for the transaction.

#### 7.11.4 Return
ZStatus_t – enum found in ZComDef.h.

### 7.12 Send Device Information Response

#### 7.12.1 Description
This function is used to send a Device Information Response.

#### 7.12.2 Prototype

```
ZStatus_t bdbTL_Send_DeviceInfoRsp (uint8 srcEP, afAddrType_t *dstAddr,
                                    bdbTLDeviceInfoRsp_t *pRsp, uint8 seqNum);
```

#### 7.12.3 Parameter Details
srcEP – The source endpoint.
dstAddr – The destination address.
pRsp– Pointer to response parameters.
seqNum - The identification number for the transaction.

#### 7.12.4 Return
ZStatus_t – enum found in ZComDef.h.

### 7.13 Send Network Start Response

#### 7.13.1 Description
This function is used to send a Network Start Response.

#### 7.13.2 Prototype

```
ZStatus_t bdbTL_Send_NwkStartRsp (uint8 srcEP, afAddrType_t *dstAddr,
                                  bdbTLNwkStartRsp_t *pRsp, uint8 seqNum);
```

#### 7.13.3 Parameter Details
srcEP – The source endpoint.
dstAddr – The destination address.
pRsp– Pointer to response parameters.
seqNum - The identification number for the transaction.

#### 7.13.4 Return
ZStatus_t – enum found in ZComDef.h.

 

## 7.14 Send Network Join Router/End Device Response

### 7.14.1 Description
This function is used to send a Network Join Router/End Device Response.

### 7.14.2 Prototype

```
bdbTL_Send_NwkJoinRsp (uint8 srcEP, afAddrType_t *dstAddr,
                       bdbTLNwkJoinRsp_t *pRsp, uint8 cmd, uint8 seqNum);
```

### 7.14.3 Parameter Details
`srcEP` – The source endpoint.
`dstAddr` – The destination address.
`pRsp`– Pointer to response parameters.
`cmd` – Join router request or join end device request.
`seqNum` - The identification number for the transaction.

### 7.14.4 Return
`ZStatus_t` – enum found in ZComDef.h.

## 7.15 Send Endpoint Information Response

### 7.15.1 Description
This function is used to send an Endpoint Information Response.

### 7.15.2 Prototype

```
ZStatus_t bdbTL_Send_EndpointInfo(uint8 srcEP, afAddrType_t *dstAddr,
        bdbTLEndpointInfo_t *pCmd, uint8 disableDefaultRsp, uint8 seqNum);
```

### 7.15.3 Parameter Details
`srcEP` – The source endpoint.
`dstAddr` – The destination address.
`pCmd` – Pinter to cmd parameters.
`disableDefaultRsp` – whether to disable the Default Response command.
`seqNum` - The identification number for the transaction.

### 7.15.4 Return
`ZStatus_t` – enum found in ZComDef.h.

## 7.16 Send Get Group Identifiers Response

### 7.16.1 Description
This function is used to send a Get Group Identifiers Response.

### 7.16.2 Prototype

```
ZStatus_t bdbTL_Send_GetGrpIDsRsp (uint8 srcEP, afAddrType_t *dstAddr,
          bdbTLGetGrpIDsRsp_t *pRsp, uint8 disableDefaultRsp, uint8 seqNum);
```

 

#### 7.16.3 Parameter Details
srcEP – The source endpoint.
dstAddr - The destination address.
pRsp– Pointer to response parameters.
disableDefaultRsp – whether to disable the Default Response command.
seqNum - The identification number for the transaction.

#### 7.16.4 Return
ZStatus_t – enum found in ZComDef.h.

### 7.17 Send Get Endpoint List Response

#### 7.17.1 Description
This function is used to send a Get Endpoint List Response.

#### 7.17.2 Prototype

```
ZStatus_t bdbTL_Send_GetEPListRsp (uint8 srcEP, afAddrType_t *dstAddr,
          bdbTLGetEPListRsp_t *pRsp, uint8 disableDefaultRsp, uint8 seqNum);
```

#### 7.17.3 Parameter Details
srcEP – The source endpoint.
dstAddr - The destination address.
pRsp– Pointer to response parameters.
disableDefaultRsp – whether to disable the Default Response command.
seqNum - The identification number for the transaction.

#### 7.17.4 Return
ZStatus_t – enum found in ZComDef.h.

### 7.18 Get Group Identifiers Request Callback

#### 7.18.1 Description
This callback is called to process an incoming Get Group Identifiers Request command.

#### 7.18.2 Prototype

```
typedef ZStatus_t (*bdbTL_GetGrpIDsReqCB_t) ( afAddrType_t *srcAddr,
                                    bdbTLGetGrpIDsReq_t *pReq, uint8 seqNum );
```

#### 7.18.3 Parameter Details
srcAddr - The destination address.
pReq– Pointer to request parameters.
seqNum - The identification number for the transaction.

#### 7.18.4 Return
ZStatus_t – enum found in ZComDef.h.

 

### 7.19 Set Endpoint List Request Callback

#### 7.19.1 Description
This callback is called to process an incoming Set Endpoint List Request command.

#### 7.19.2 Prototype

```
typedef ZStatus_t (*bdbTL_GetEPListReqCB_t)( afAddrType_t *srcAddr,
                                    bdbTLGetEPListReq_t *pReq, uint8 SeqNum );
```

#### 7.19.3 Parameter Details
`srcAddr` – The destination address.
`pReq`– Pointer to request parameters.
`seqNum` - The identification number for the transaction.

#### 7.19.4 Return
`ZStatus_t` – enum found in ZComDef.h.

### 7.20 Endpoint Information Callback

#### 7.20.1 Description
This callback is called to process an incoming Endpoint Information command.

#### 7.20.2 Prototype

```
typedef ZStatus_t (*bdbTL_EndpointInfoCB_t)( afAddrType_t *srcAddr,
                                               bdbTLEndpointInfo_t *pRsp );
```

#### 7.20.3 Parameter Details
`srcAddr` – The destination address.
`pReq`– Pointer to request parameters.
`seqNum` - The identification number for the transaction.

#### 7.20.4 Return
`ZStatus_t` – enum found in ZComDef.h.

### 7.21 Get Group Identifiers Callback

#### 7.21.1 Description
This callback is called to process an incoming Get Group Identifiers command.

#### 7.21.2 Prototype

```
typedef ZStatus_t (*bdbTL_GetGrpIDsRspCB_t)( afAddrType_t *srcAddr,
                                                bdbTLGetGrpIDsRsp_t *pRsp );
```

#### 7.21.3 Parameter Details
`srcAddr` – The destination address.
`pReq`– Pointer to request parameters.
`seqNum` - The identification number for the transaction.

#### 7.21.4 Return
`ZStatus_t` – enum found in ZComDef.h.

 

### 7.22 Get Endpoint List Response Callback

#### 7.22.1 Description
This callback is called to process an incoming Get Endpoint List Response command.

#### 7.22.2 Prototype

```
typedef ZStatus_t (*bdbTL_GetEPListRspCB_t)( afAddrType_t *srcAddr,
                                             bdbTLGetEPListRsp_t *pRsp );
```

#### 7.22.3 Parameter Details
`srcAddr` - The destination address.
`pReq`– Pointer to request parameters.
`seqNum` - The identification number for the transaction.

#### 7.22.4 Return
`ZStatus_t` – enum found in ZComDef.h.

### 7.23 Scan Request Callback

#### 7.23.1 Description
This callback is called to process an incoming Scan Request command.

#### 7.23.2 Prototype

```
typedef ZStatus_t (*bdbTL_ScanReqCB_t)( afAddrType_t *srcAddr,
                                        bdbTLScanReq_t *pReq, uint8 seqNum );
```

#### 7.23.3 Parameter Details
`srcAddr` - The destination address.
`pReq`– Pointer to request parameters.
`seqNum` - The identification number for the transaction.

#### 7.23.4 Return
`ZStatus_t` – enum found in ZComDef.h.

### 7.24 Device Information Request Callback

#### 7.24.1 Description
This callback is called to process an incoming Device Information Request command.

#### 7.24.2 Prototype

```
typedef ZStatus_t (*bdbTL_DeviceInfoReqCB_t)( afAddrType_t *srcAddr,
                                    bdbTLDeviceInfoReq_t *pReq, uint8 seqNum );
```

#### 7.24.3 Parameter Details
`srcAddr` - The destination address.
`pReq`– Pointer to request parameters.
`seqNum` - The identification number for the transaction.

#### 7.24.4 Return
`ZStatus_t` – enum found in ZComDef.h.

 

### 7.25 Identify Request Callback

#### 7.25.1 Description
This callback is called to process an incoming Identify Request command.

#### 7.25.2 Prototype

```
typedef ZStatus_t (*bdbTL_IdentifyReqCB_t)( afAddrType_t *srcAddr,
                                            bdbTLIdentifyReq_t *pReq );
```

#### 7.25.3 Parameter Details
`srcAddr` – The destination address.
`pReq`– Pointer to request parameters.
`seqNum` - The identification number for the transaction.

#### 7.25.4 Return
`ZStatus_t` – enum found in ZComDef.h.

### 7.26 Reset to Factory New Request Callback

#### 7.26.1 Description
This callback is called to process an incoming Reset to Factory New Request command.

#### 7.26.2 Prototype

```
typedef ZStatus_t (*bdbTL_ResetToFNReqCB_t)( afAddrType_t *srcAddr,
                                             bdbTLResetToFNReq_t *pReq );
```

#### 7.26.3 Parameter Details
`srcAddr` – The destination address.
`pReq`– Pointer to request parameters.
`seqNum` - The identification number for the transaction.

#### 7.26.4 Return
`ZStatus_t` – enum found in ZComDef.h.

### 7.27 Network Start Request Callback

#### 7.27.1 Description
This callback is called to process an incoming Network Start Request command.

#### 7.27.2 Prototype

```
typedef ZStatus_t (*bdbTL_NwkStartReqCB_t)( afAddrType_t *srcAddr,
                                    bdbTLNwkStartReq_t *pReq, uint8 seqNum );
```

#### 7.27.3 Parameter Details
`srcAddr` – The destination address.
`pReq`– Pointer to request parameters.
`seqNum` - The identification number for the transaction.

#### 7.27.4 Return
`ZStatus_t` – enum found in ZComDef.h.

 

### 7.28 Network Join Router Request Callback

#### 7.28.1 Description
This callback is called to process an incoming Network Join Router Request command.

#### 7.28.2 Prototype

```
typedef ZStatus_t (*bdbTL_NwkJoinRtrReqCB_t)( afAddrType_t *srcAddr,
                                        bdbTLNwkJoinReq_t *pReq, uint8 seqNum );
```

#### 7.28.3 Parameter Details
`srcAddr` – The destination address.
`pReq`– Pointer to request parameters.
`seqNum` - The identification number for the transaction.

#### 7.28.4 Return
`ZStatus_t` – enum found in ZComDef.h.

### 7.29 Network Join End Device Request Callback

#### 7.29.1 Description
This callback is called to process an incoming Network Join End Device Request command.

#### 7.29.2 Prototype

```
typedef ZStatus_t (*bdbTL_NwkJoinEDReqCB_t)( afAddrType_t *srcAddr,
                                        bdbTLNwkJoinReq_t *pReq, uint8 seqNum );
```

#### 7.29.3 Parameter Details
`srcAddr` – The destination address.
`pReq`– Pointer to request parameters.
`seqNum` - The identification number for the transaction.

#### 7.29.4 Return
`ZStatus_t` – enum found in ZComDef.h.

### 7.30 Network Update Request Callback

#### 7.30.1 Description
This callback is called to process an incoming Network Update Request command.

#### 7.30.2 Prototype

```
typedef ZStatus_t (*bdbTL_NwkUpdateReqCB_t)( afAddrType_t *srcAddr,
                                             bdbTLNwkUpdateReq_t *pReq );
```

#### 7.30.3 Parameter Details
`srcAddr` – The destination address.
`pReq`– Pointer to request parameters.

#### 7.30.4 Return
`ZStatus_t` – enum found in ZComDef.h.

 

### 7.31 Scan Response Callback

#### 7.31.1 Description
This callback is called to process an incoming Scan Response command.

#### 7.31.2 Prototype

```
typedef ZStatus_t (*bdbTL_ScanRspCB_t)( afAddrType_t *srcAddr,
                                        bdbTLScanRsp_t *pRsp );
```

#### 7.31.3 Parameter Details
`srcAddr` - The destination address.
`pRsp`– Pointer to response parameters.

#### 7.31.4 Return
`ZStatus_t` – enum found in ZComDef.h.

### 7.32 Device Information Response Callback

#### 7.32.1 Description
This callback is called to process an incoming Device Information Response command.

#### 7.32.2 Prototype

```
typedef ZStatus_t (*bdbTL_DeviceInfoRspCB_t)( afAddrType_t *srcAddr,
                                              bdbTLDeviceInfoRsp_t *pRsp );
```

#### 7.32.3 Parameter Details
`srcAddr` - The destination address.
`pRsp`– Pointer to response parameters.

#### 7.32.4 Return
`ZStatus_t` – enum found in ZComDef.h.

### 7.33 Network Start Response Callback

#### 7.33.1 Description
This callback is called to process an incoming Network Start Response command.

#### 7.33.2 Prototype

```
typedef ZStatus_t (*bdbTL_NwkStartRspCB_t)( afAddrType_t *srcAddr,
                                            bdbTLNwkStartRsp_t *pRsp );
```

#### 7.33.3 Parameter Details
`srcAddr` - The destination address.
`pRsp`– Pointer to response parameters.

#### 7.33.4 Return
`ZStatus_t` – enum found in ZComDef.h.

 

### 7.34 Network Join Router Response Callback

#### 7.34.1 Description
This callback is called to process an incoming Network Join Router Response command.

#### 7.34.2 Prototype

```
typedef ZStatus_t (*bdbTL_NwkJoinRtrRspCB_t)( afAddrType_t *srcAddr,
                                              bdbTLNwkJoinRsp_t *pRsp );
```

#### 7.34.3 Parameter Details
`srcAddr` - The destination address.
`pRsp`– Pointer to response parameters.

#### 7.34.4 Return
`ZStatus_t` – enum found in ZComDef.h.

### 7.35 Network Join End Device Response Callback

#### 7.35.1 Description
This callback is called to process an incoming Network Join End Device Response command.

#### 7.35.2 Prototype

```
typedef ZStatus_t (*bdbTL_NwkJoinEDRspCB_t)( afAddrType_t *srcAddr,
                                             bdbTLNwkJoinRsp_t *pRsp );
```

#### 7.35.3 Parameter Details
`srcAddr` - The destination address.
`pRsp`– Pointer to response parameters.

#### 7.35.4 Return
`ZStatus_t` – enum found in ZComDef.h.

 

# 8. Green Power

The current implementation of Green Power endpoint is to provide basic proxy functionality, which is meant to relay packets from the Green Power Device to a Green Power Sink device after a commissioning process performed by the Green Power Device and the Green Power Sink. This application is already managed by the stack and it does not required user interaction. The API here stated is provided for such applications seeking to expand the Green Power functionality to its application.

## 8.1 Register Green Power Application Command Callbacks

### 8.1.1 Description

This function is used to send a Green Power Notification.

### 8.1.2 Prototype

```
ZStatus_t zclGp_RegisterCmdCallbacks (uint8 endpoint,
                                      zclGp_AppCallbacks_t *callbacks);
```

### 8.1.3 Parameter Details

`endpoint` – Target application endpoint.
`callbacks` – Pointer to the callback record.

### 8.1.4 Return

`ZStatus_t` – enum found in ZComDef.h.

## 8.2 Send Green Power Notification

### 8.2.1 Description

This function is used to send a Green Power Notification.

### 8.2.2 Prototype

```
ZStatus_t zclGp_SendGpNotificationCommand( gpNotificationCmd_t *pCmd );
```

### 8.2.3 Parameter Details

`pCmd` – Pointer to Green Power Notification Command data.

### 8.2.4 Return

`ZStatus_t` – enum found in ZComDef.h.

## 8.3 Send Green Power Commissioning Notification

### 8.3.1 Description

This function is used to send a Green Power Commissioning Notification.

### 8.3.2 Prototype

```
ZStatus_t zclGp_SendGpCommissioningNotificationCommand
                                  (gpCommissioningNotificationCmd_t *pCmd);
```

### 8.3.3 Parameter Details

`pCmd` – Pointer to Green Power Commissioning Notification Command data.

 

#### 8.3.4 Return
ZStatus_t – enum found in ZComDef.h.

### 8.4 Send Green Power Proxy Table Request

#### 8.4.1 Description
This function is used to send a Green Power Proxy Table Request.

#### 8.4.2 Prototype

```
ZStatus_t zclGp_SendGpProxyTableResponse (afAddrType_t *dstAddr,
                                zclGpProxyTableResponse_t *rsp, uint8 seqNum);
```

#### 8.4.3 Parameter Details
dstAddr - The destination address.
pRsp– Pointer to response parameters.
seqNum - The identification number for the transaction.

#### 8.4.4 Return
ZStatus_t – enum found in ZComDef.h.

### 8.5 Gp Pairing Callback

#### 8.5.1 Description
This callback is called to process an incoming Gp Pairing command.

#### 8.5.2 Prototype

```
typedef void (*zclGCB_GP_Pairing_t)( zclGpPairing_t *pCmd );
```

#### 8.5.3 Parameter Details
pCmd– Pointer to command parameters.

#### 8.5.4 Return
None.

### 8.6 Gp Commissioning Mode Callback

#### 8.6.1 Description
This callback is called to process an incoming Gp Commissioning Mode command.

#### 8.6.2 Prototype

```
typedef void (*zclGCB_GP_Proxy_Commissioning_Mode_t)
                                      ( zclGpProxyCommissioningMode_t *pCmd );
```

#### 8.6.3 Parameter Details
pCmd– Pointer to command parameters.

#### 8.6.4 Return
None.

 

## 8.7 Gp Response Callback

### 8.7.1 Description
This callback is called to process an incoming Gp Commissioning Mode command.

### 8.7.2 Prototype

```
typedef void (*zclGCB_GP_Response_t)( zclGpResponse_t *pCmd );
```

### 8.7.3 Parameter Details
pCmd– Pointer to command parameters.

### 8.7.4 Return
None.

## 8.8 Gp Response Callback

### 8.8.1 Description
This callback is called to process an incoming Gp Commissioning Mode command.

### 8.8.2 Prototype

```
typedef void (*zclGCB_GP_Proxy_Table_Request_t)
                                        ( zclGpProxyTableRequest_t *pReq );
```

### 8.8.3 Parameter Details
pReq– Pointer to request parameters.

### 8.8.4 Return
None.

 

## 9. Compile Options

The ZCL compile options are defined in the ZCL configuration file *f8wZCL.cfg*, which is located in the **Tools** folder of the Z-Stack installation, along with other configuration files. The *f8wZCL.cfg* file is used by all projects that include the ZCL (i.e., all Home Automation projects). Therefore, any change made to this file will affect all HA projects. Compile options for other profiles, such as Smart Energy [5] , can be found in their API documents. If needed, you can create a private version of the *f8wZCL.cfg* file and modify your project to use the new version. The ZCL supported compile options and their definitions are listed in the following table:

| | |
|---|---|
| **ZCL_READ** | Enable the following commands:<br>1) Read Attributes<br>2) Read Attributes Response |
| **ZCL_WRITE** | Enable the following commands:<br>1) Write Attributes<br>2) Write Attributes Undivided<br>3) Write Attributes Response<br>4) Write Attributes No Response |
| **ZCL_REPORTING_DEVICE** | Enable the following commands:<br>1) Configure Reporting Response<br>2) Report Attributes |
| **ZCL_REPORT_CONFIGURING_DEVICE** | Enable the following commands:<br>1) Configure Reporting |
| **ZCL_REPORT_DESTINATION_DEVICE** | Enable reception of the following commands<br>1) Read Reporting Response<br>2) Read Reporting Configuration Response |
| **ZCL_DISCOVER** | Enable the following commands:<br>1) Discover Attributes<br>2) Discover Attributes Response |
| **ZCL_BASIC** | Enable the following command:<br>1) Reset to Factory Defaults |
| **ZCL_IDENTIFY** | Enable the following commands:<br>1) Identify<br>2) Identify Query<br>3) Identify Query Response |
| **ZCL_GROUPS** | Enable the following commands:<br>1) Add Group<br>2) View Group<br>3) Get Group Membership<br>4) Remove Group<br>5) Remove All Groups<br>6) Add Group If Identifying<br>7) Add Group Response<br>8) View Group Response<br>9) Get Group Membership Response<br>10) Remove Group Response |
| **ZCL_SCENES** | Enable the following commands:<br>1) Add Scene<br>2) View Scene<br>3) Remove Group<br>4) Remove All Groups<br>5) Store Scene |

 

| | |
|---|---|
| | 6) Recall Scene<br>7) Get Scene Membership<br>8) Add Scene Response<br>9) View Scene Response<br>10) Remove Scene Response<br>11) Remove All Scenes Response<br>12) Store Scene Response<br>13) Get Scene Membership Response |
| **ZCL_ON_OFF** | Enable the following commands:<br>1) On<br>2) Off<br>3) Toggle |
| **ZCL_LEVEL_CTRL** | Enable the following commands:<br>1) Move to Level<br>2) Move<br>3) Step<br>4) Stop<br>5) Move to Level with On/Off<br>6) Move with On/Off<br>7) Step with On/Off<br>8) Stop with On/Off |
| **ZCL_ALARMS** | Enable the following commands:<br>1) Reset Alarm<br>2) Reset All Alarms<br>3) Get Alarm<br>4) Reset Alarm Log<br>5) Alarm<br>6) Get Alarm Response |
| **ZCL_LOCATION** | Enable the following commands:<br>1) Set Absolute Location<br>2) Set Device Configuration<br>3) Get Device Configuration<br>4) Get Location Data<br>5) Device Configuration Response<br>6) Location Data Response<br>7) Location Data Notification<br>8) Compact Location Data Notification<br>9) RSSI Ping |
| **ZCL_ZONE** | Enable the following commands:<br>1) Zone Status Change Notification<br>2) Zone Enroll Request<br>3) Zone Enroll Response |
| **ZCL_ACE** | Enable the following commands:<br>1) Arm<br>2) Bypass<br>3) Emergency<br>4) Fire<br>5) Panic<br>6) Get Zone ID Map<br>7) Get Zone Information<br>8) Arm Response<br>9) Get Zone ID Map Response<br>10) Get Zone Information Response |

 

| | |
|---|---|
| **ZCL_WD** | Enable the following commands:<br>1) Start Warning<br>2) Squawk |
| **ZCL_DOORLOCK** | Enable the following commands:<br>1) Door Lock<br>2) Door Lock Response<br>3) Door Unlock<br>4) Door Unlock Response<br>5) Door Toggle<br>5) Door Toggle Response |
| **ZCL_DOORLOCK_EXT** | Enable the following commands and its respective responses:<br>1) UnlockWithTimeout<br>2) GetLogRecord<br>3) SetPINCode<br>4) GetPINCode<br>5) ClearPINCode<br>6) ClearAllPINCodes<br>7) SetUserStatus<br>8) GetUserStatus<br>9) SetWeekDaySchedule<br>10) GetWeekDaySchedule<br>11) ClearWeekDaySchedule<br>12) SetYearDaySchedule<br>13) GetYearDaySchedule<br>14) ClearYearDaySchedule<br>15) SetHolidaySchedule<br>16) GetHolidaySchedule<br>17) ClearHolidaySchedule<br>18) SetUserType<br>19) GetUserType<br>20) SetRFIDCode<br>21) GetRFIDCode<br>22) ClearRFIDCode<br>23) ClearAllRFIDCodes<br>24) OperationEventNotification<br>25) ProgrammingEventNotification |
| **ZCL_WINDOWCOVERING** | Enable the following Window Covering Cluster commands:<br>1) Up / Open<br>2) Down / Close<br>3) Stop<br>4) Go to Lift Setpoint<br>5) Go to Lift Value<br>6) Go to Lift Percentage<br>7) Go to Tilt Setpoint<br>8) Go to Tilt Value<br>9) Go to Tilt Percentage<br>10) Program Setpoint |

 