



Z-Stack

ZigBee Cluster Library (ZCL)

Application Programming Interface

Document Number: F8W-2006-0020

Texas Instruments, Inc.
San Diego, California USA
(619) 497-3845

Version	Description	Date
1.0	Initial release.	12/7/2006
1.1	Added Compile Options	3/20/2007
1.2	Updated as per latest specs (Foundation 06027r06 and General 053936r05)	5/2/2007
1.3	Updated as per latest spec (075123r01ZB)	11/28/2007

TABLE OF CONTENTS

1. INTRODUCTION.....	1
1.1 PURPOSE	1
1.2 SCOPE	1
1.3 ACRONYMS	1
1.4 APPLICABLE DOCUMENTS	1
2. API OVERVIEW	2
2.1 OVERVIEW	2
2.2 CLIENT/SERVER MODEL	2
2.3 STACK DIAGRAM	2
2.4 APPLICATION/PROFILE REGISTRATION	3
2.5 APPLICATION CREATION	4
3. FOUNDATION LAYER	6
3.1 INTRODUCTION	6
3.2 SEND COMMAND.....	6
3.3 SEND READ	7
3.4 SEND READ RESPONSE.....	8
3.5 SEND WRITE	8
3.6 SEND WRITE UNDIVIDED	9
3.7 SEND WRITE RESPONSE	10
3.8 SEND WRITE NO RESPONSE	10
3.9 SEND CONFIGURE REPORTING	11
3.10 SEND CONFIGURE REPORTING RESPONSE	11
3.11 SEND READ REPORTING CONFIGURATION.....	12
3.12 SEND READ REPORTING CONFIGURATION RESPONSE	13
3.13 SEND REPORT.....	13
3.14 SEND DEFAULT RESPONSE	14
3.15 SEND DISCOVER	14
3.16 SEND DISCOVER RESPONSE.....	15
3.17 REGISTER ATTRIBUTE LIST	16
3.18 REGISTER ATTRIBUTE DATA VALIDATION CALLBACK	16
3.19 REGISTER CLUSTER LIBRARY HANDLER CALLBACK.....	16
3.20 CLUSTER LIBRARY HANDLER CALLBACK	17
3.21 REGISTER CLUSTER CONVERT TABLE	17
4. GENERAL FUNCTIONAL DOMAIN	18
4.1 INTRODUCTION	18
4.2 SEND RESET TO FACTORY DEFAULTS (BASIC).....	18
4.3 SEND IDENTIFY (IDENTIFY).....	19
4.4 SEND IDENTIFY QUERY (IDENTIFY).....	19
4.5 SEND IDENTIFY QUERY RESPONSE (IDENTIFY)	20
4.6 SEND ADD GROUP (GROUP).....	20
4.7 SEND VIEW GROUP (GROUP)	21
4.8 SEND GET GROUP MEMBERSHIP (GROUP)	21
4.9 SEND REMOVE GROUP (GROUP)	22
4.10 SEND REMOVE ALL GROUPS (GROUP)	23
4.11 SEND ADD GROUP IF IDENTIFYING (GROUP).....	23
4.12 SEND ADD GROUP RESPONSE (GROUP).....	24
4.13 SEND VIEW GROUP RESPONSE (GROUP)	24
4.14 SEND GET GROUP MEMBERSHIP RESPONSE (GROUP)	25
4.15 SEND REMOVE GROUP RESPONSE (GROUP)	25
4.16 SEND ADD SCENE (SCENE).....	26

4.17	SEND VIEW SCENE (SCENE)	26
4.18	SEND REMOVE SCENE (SCENE)	27
4.19	SEND REMOVE ALL SCENES (SCENE)	28
4.20	SEND STORE SCENE (SCENE).....	28
4.21	SEND RECALL SCENE (SCENE)	29
4.22	SEND GET SCENE MEMBERSHIP (SCENE)	29
4.23	SEND ADD SCENE RESPONSE (SCENE).....	30
4.24	SEND VIEW SCENE RESPONSE (SCENE)	30
4.25	SEND REMOVE SCENE RESPONSE (SCENE)	31
4.26	SEND REMOVE ALL SCENES (SCENE).....	32
4.27	SEND STORE SCENE (SCENE).....	32
4.28	SEND GET SCENE MEMBERSHIP RESPONSE (SCENE)	33
4.29	SEND OFF (ON/OFF)	33
4.30	SEND ON (ON/OFF)	34
4.31	SEND TOGGLE (ON/OFF)	34
4.32	SEND MOVE TO LEVEL (LEVEL CONTROL).....	35
4.33	SEND MOVE (LEVEL CONTROL)	35
4.34	SEND STEP (LEVEL CONTROL)	36
4.35	SEND RESET ALARM (ALARM)	37
4.36	SEND RESET ALL ALARMS (ALARM).....	37
4.37	SEND GET ALARM (ALARM)	38
4.38	SEND RESET ALARM LOG (ALARM)	38
4.39	SEND ALARM (ALARM).....	39
4.40	SEND GET ALARM RESPONSE (ALARM)	39
4.41	SEND SET ABSOLUTE LOCATION (RSSI LOCATION).....	40
4.42	SEND SET DEVICE CONFIGURATION (RSSI LOCATION).....	40
4.43	SEND GET DEVICE CONFIGURATION (RSSI LOCATION)	41
4.44	SEND GET LOCATION DATA (RSSI LOCATION).....	42
4.45	SEND DEVICE CONFIGURATION RESPONSE (RSSI LOCATION)	42
4.46	SEND LOCATION DATA RESPONSE (RSSI LOCATION)	43
4.47	SEND LOCATION DATA NOTIFICATION (RSSI LOCATION)	43
4.48	SEND COMPACT LOCATION DATA NOTIFICATION (RSSI LOCATION).....	44
4.49	SEND RSSI PING (RSSI LOCATION)	44
4.50	ATTRIBUTE DATA VALIDATION CALLBACK.....	45
4.51	REGISTER APPLICATION COMMAND CALLBACK	45
4.52	RESET TO FACTORY DEFAULTS CALLBACK.....	46
4.53	IDENTIFY CALLBACK.....	46
4.54	IDENTIFY RESPONSE CALLBACK	47
4.55	ON/OFF/TOGGLE CALLBACK.....	47
4.56	MOVE TO LEVEL CALLBACK.....	47
4.57	MOVE CALLBACK	48
4.58	STEP CALLBACK.....	48
4.59	STEP CALLBACK.....	49
4.60	GROUP RESPONSE CALLBACK.....	49
4.61	STORE SCENE CALLBACK.....	50
4.62	RECALL SCENE CALLBACK	50
4.63	SCENE RESPONSE CALLBACK.....	51
4.64	ALARM CALLBACK	52
4.65	LOCATION CALLBACK.....	52
4.66	LOCATION RESPONSE CALLBACK.....	53
5.	COMPILE OPTIONS.....	53

LIST OF FIGURES

FIGURE 1: STACK DIAGRAM.....3

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 a number of 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

This document covers only the Foundation layer and the General functional domain.

1.2 Scope

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

1.3 Acronyms

AF	Application Framework
API	Application Programming Interface
APS	Application Support Sub-Layer
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
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
NWK	Network Layer
PAN	Personal Area Network
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

2. API Overview

2.1 Overview

The ZCL acts as a repository for cluster functionality that is developed by ZigBee. A developer constructing a new profile should incorporate relevant ZCL cluster functionality into the new profile. 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 Profile's real-to-logical cluster ID conversion table

The General functional domain provides 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.

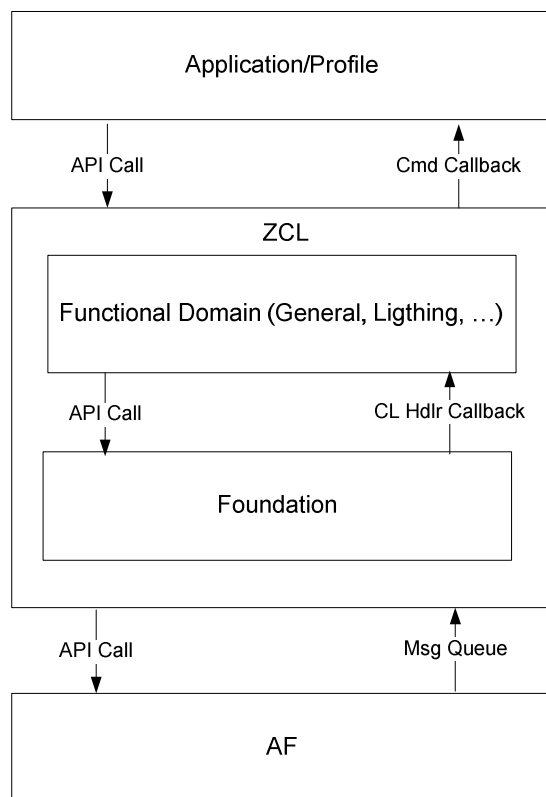


Figure 1: Stack Diagram

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_registerClusterConvertTable()` APIs to the Application/Profile to register Application's attribute list, attribute data validation callback and Profile's real-to-logical cluster ID conversion table 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;    // Real cluster ID
    zclAttribute_t  attr;
} zclAttrRec_t;

```

The cluster list input parameter to `zcl_registerClusterConvertTable()` contains the actual and logical cluster IDs for each supported cluster:

```

/*****
 * Cluster ID Conversion table - since the cluster IDs are assigned by
 * the profile and internally we use logical cluster IDs
 */
typedef struct
{
    uint16  actualCluster;
    uint16  logicalCluster;
} zclConvertClusterRec_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;        // Basic Reset
    zclGCB_Identify_t            pfnIdentify;          // Identify Response
    zclGCB_IdentifyQueryRsp_t    pfnIdentifyQueryRsp; // Identify Query Rsp
    zclGCB_OnOff_t               pfnOnOff;            // On/Off cluster
    zclGCB_LevelControlMoveToLevel_t pfnLevelControlMoveToLevel; // MoveToLevel
    zclGCB_LevelControlMove_t    pfnLevelControlMove; // Move
    zclGCB_LevelControlStep_t    pfnLevelControlStep; // Step
    zclGCB_LevelControlStop_t    pfnLevelControlStop; // Stop
    zclGCB_GroupRsp_t            pfnGroupRsp;         // Group Response
    zclGCB_SceneStoreReq_t       pfnSceneStoreReq;    // Scene Store Request
    zclGCB_SceneRecallReq_t      pfnSceneRecallReq;   // Scene Recall Request
    zclGCB_SceneRsp_t            pfnSceneRsp;         // Scene Response
    zclGCB_Alarm_t               pfnAlarm;           // Alarm Req & Rsp
    zclGCB_Location_t            pfnLocation;         // RSSI Location
    zclGCB_LocationRsp_t         pfnLocationRsp;      // RSSI Location Rsp
} zclGeneral_AppCallbacks_t;

```

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

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 `zclHA_Init()` API defined in `zcl_ha.c` module

2.5.4 OSAL_<AppName>.c

This module should contain `void osalAddTasks(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 `osalTaskAdd()` API defined in `OSAL_Tasks.c` module. Here's the minimum list of tasks and their addition order needed for a simple ZCL application:

1. HAL
2. MAC
3. Network
4. APS
5. ZD Application
6. ZCL
7. ZCL Application

Note: The ZCL task should be added first before any ZCL application task.

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 logicalCluster,  
                           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.

logicalCluster - Whether the cluster ID is a logical cluster ID.

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 realClusterID, zclReadCmd_t *readCmd,  
                        uint8 direction, uint8 disableDefaultRsp,  
                        uint8 seqNum );
```

3.3.3 Parameter Details

srcEP – The source endpoint.

destAddr – The destination address.

`realClusterID` - 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 realClusterID, zclReadRspCmd_t *readRspCmd,  
                           uint8 direction, uint8 disableDefaultRsp,  
                           uint8 seqNum );
```

3.4.3 Parameter Details

`srcEP` - The source endpoint.
`destAddr` - The destination address.
`realClusterID` - 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 realClusterID, zclWriteCmd_t *writeCmd,  
                        uint8 direction, uint8 disableDefaultRsp,  
                        uint8 seqNum );
```

3.5.3 Parameter Details

srcEP – The source endpoint.

destAddr – The destination address.

realClusterID – 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 realClusterID, zclWriteCmd_t *writeCmd,  
                                  uint8 direction, uint8 disableDefaultRsp,  
                                  uint8 seqNum );
```

3.6.3 Parameter Details

srcEP – The source endpoint.

destAddr – The destination address.

realClusterID – 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 realClusterID, zclWriteRspCmd_t *writeRspCmd,  
                           uint8 direction, uint8 disableDefaultRsp,  
                           uint8 seqNum );
```

3.7.3 Parameter Details

srcEP – The source endpoint.

destAddr – The destination address.

realClusterID – 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 realClusterID, zclWriteCmd_t *writeCmd,  
                              uint8 direction, uint8 disableDefaultRsp,  
                              uint8 seqNum );
```

3.8.3 Parameter Details

srcEP – The source endpoint.

destAddr – The destination address.

`realClusterID` - 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 realClusterID, zclCfgReportCmd_t *cfgReportCmd,  
                                   uint8 direction, uint8 disableDefaultRsp, uint8 seqNum );
```

3.9.3 Parameter Details

`srcEP` - The source endpoint.
`destAddr` - The destination address.
`realClusterID` - 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 realClusterID, zclCfgReportRspCmd_t *cfgReportRspCmd,  
                                     uint8 direction, uint8 disableDefaultRsp, uint8 seqNum );
```

3.10.3 Parameter Details

srcEP – The source endpoint.

destAddr – The destination address.

realClusterID – 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 realClusterID, zclReadReportCfgCmd_t *readReportCfgCmd,  
                                    uint8 direction, uint8 disableDefaultRsp, uint8 seqNum );
```

3.11.3 Parameter Details

srcEP – The source endpoint.

destAddr – The destination address.

realClusterID – 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 realClusterID, zclReadReportCfgRspCmd_t *readReportCfgRspCmd,  
                                       uint8 direction, uint8 disableDefaultRsp, uint8 seqNum );
```

3.12.3 Parameter Details

srcEP – The source endpoint.

destAddr – The destination address.

realClusterID – 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 realClusterID, zclReportCmd_t *reportCmd,  
                             uint8 direction, uint8 disableDefaultRsp,  
                             uint8 seqNum );
```

3.13.3 Parameter Details

srcEP – The source endpoint.

destAddr – The destination address.

realClusterID – 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 realClusterID, zclDefaultRspCmd_t *defaultRspCmd,  
                                uint8 direction, uint8 disableDefaultRsp, uint8 seqNum );
```

3.14.3 Parameter Details

srcEP – The source endpoint.

destAddr – The destination address.

realClusterID – 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 realClusterID, zclDiscoverCmd_t *discoverCmd,  
                               uint8 direction, uint8 disableDefaultRsp,  
                               uint8 seqNum );
```

3.15.3 Parameter Details

srcEP – The source endpoint.

destAddr – The destination address.

realClusterID – 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 realClusterID, zclDiscoverRspCmd_t *discoverRspCmd,  
                                  uint8 direction, uint8 disableDefaultRsp, uint8 seqNum );
```

3.16.3 Parameter Details

srcEP – The source endpoint.

destAddr – The destination address.

realClusterID – 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 startLogCluster,  
                             uint16 endLogCluster,  
                             zclInHdlr_t pfnIncomingHdlr );
```

3.19.3 Parameter Details

startLogCluster – The starting logical cluster ID.

endLogCluster – The ending logical 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,  
                                  uint16 logicalClusterID );
```

3.20.3 Parameter Details

msg – The incoming message.

logicalClusterID – The logical cluster ID.

3.20.4 Return

ZStatus_t – enum found in ZComDef.h.

3.21 Register Cluster Convert Table

3.21.1 Description

This function is used to register a Profile's Cluster Conversion Table with the ZCL Foundation layer.

3.21.2 Prototype

```
ZStatus_t zcl_registerClusterConvertTable( uint16 profileId,  
                                           uint16 numClusters,
```

```
zclConvertClusterRec_t GENERIC *clusterList);
```

3.21.3 Parameter Details

`profileId` – The profile ID.

`numClusters` – The number of clusters in the list.

`clusterList` – The Profile's Cluster Conversion List.

3.21.4 Return

`ZStatus_t` – enum found in `ZComDef.h`.

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.

dstAddr – 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.

dstAddr – 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.

dstAddr – 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.

dstAddr – 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.

dstAddr – 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.

dstAddr – 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.

dstAddr – 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.

dstAddr – 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.

dstAddr – 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.

dstAddr – 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.

dstAddr – 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.

dstAddr – 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.

dstAddr - 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.

dstAddr – 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.

dstAddr – 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.

dstAddr – 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.

dstAddr – 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.

dstAddr – 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.

dstAddr – 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.

dstAddr – 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.

dstAddr – 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.

dstAddr – 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.

dstAddr – 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.

dstAddr – 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.

dstAddr - 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.

dstAddr – 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.

dstAddr – 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.

dstAddr – 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.

dstAddr – 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.

dstAddr – 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.

dstAddr – 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.

dstAddr – 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.

dstAddr – 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.

dstAddr – 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.

dstAddr – 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.

dstAddr – 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_SendAlarmGetResponse( 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.

dstAddr – 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.

dstAddr – 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.

dstAddr – 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.

dstAddr – 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.

dstAddr – 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_SendLocationDataNotif( 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.

dstAddr – 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)( zclWriteRec_t *pAttrInfo );
```

4.50.3 Parameter Details

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)( afAddrType_t *dstAddr,  
                                  uint16 identifyTime);
```

4.53.3 Parameter Details

`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)( afAddrType_t *dstAddr,  
                                     uint16 timeout );
```

4.54.3 Parameter Details

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 – The 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)( uint8 level,  
                                                  uint16 transitionTime,  
                                                  uint8 withOnOff );
```

4.56.3 Parameter Details

`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)( uint8 moveMode, uint8 rate,  
                                           uint8 withOnOff );
```

4.57.3 Parameter Details

`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)( uint8 stepMode,  
                                           uint8 amount,  
                                           uint16 transitionTime,  
                                           uint8 withOnOff );
```

4.58.3 Parameter Details

`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)( afAddrType_t *srcAddr, uint8 cmdID,  
                                  uint8 status, uint8 grpCnt,  
                                  uint16 *grpList, uint8 capacity,  
                                  uint8 *grpName );
```

4.60.3 Parameter Details

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)( afAddrType_t *srcAddr,  
                                         zclGeneral_Scene_t *scene );
```

4.61.3 Parameter Details

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)( afAddrType_t *srcAddr,
                                         zclGeneral_Scene_t *scene );
```

4.62.3 Parameter Details

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)( afAddrType_t *srcAddr, uint8 cmdID,
                                   uint8 status, uint8 sceneCnt,
                                   uint8 *sceneList, uint8 capacity,
                                   zclGeneral_Scene_t *scene );
```

4.63.3 Parameter Details

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)( afAddrType_t *srcAddr, uint8 cmdID,  
                                uint8 status, uint8 alarmCode,  
                                uint16 clusterID, uint32 timeStamp );
```

4.64.3 Parameter Details

`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)( afAddrType_t *srcAddr, uint8 cmdID,  
                                   zclLocationAbsolute_t *absLoc,  
                                   zclLocationGetData_t *loc,  
                                   zclLocationDevCfg_t *devCfg,  
                                   uint8 *ieeeAddr, uint8 seqNum );
```

4.65.3 Parameter Details

`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)( afAddrType_t *srcAddr, uint8 cmdID,
                                     zclLocationDataRsp_t *locRsp,
                                     zclLocationDevCfgRsp_t *devCfgRsp,
                                     uint8 locationType );
```

4.66.3 Parameter Details

`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.

5. Compile Options

The ZCL compile options are defined in the ZCL linker control file `f8wZCL.cfg`, which is located in the **Tools** folder along with other linker control 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. 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: 1) Read Attributes 2) Read Attributes Response
ZCL_WRITE	Enable the following commands: 1) Write Attributes 2) Write Attributes Undivided 3) Write Attributes Response 4) Write Attributes No Response
ZCL_REPORT	Enable the following commands: 1) Configure Reporting 2) Configure Reporting Response 3) Read Reporting Response 4) Read Reporting Configuration Response 5) Report Attributes
ZCL_DISCOVER	Enable the following commands: 1) Discover Attributes 2) Discover Attributes Response
ZCL_BASIC	Enable the following command: 1) Reset to Factory Defaults
ZCL_IDENTIFY	Enable the following commands: 1) Identify 2) Identify Query 3) Identify Query Response
ZCL_GROUPS	Enable the following commands: 1) Add Group 2) View Group 3) Get Group Membership 4) Remove Group 5) Remove All Groups 6) Add Group If Identifying 7) Add Group Response 8) View Group Response 9) Get Group Membership Response 10) Remove Group Response
ZCL_SCENES	Enable the following commands: 1) Add Scene 2) View Scene 3) Remove Group 4) Remove All Groups 5) Store Scene 6) Recall Scene 7) Get Scene Membership 8) Add Scene Response 9) View Scene Response 10) Remove Scene Response 11) Remove All Scenes Response 12) Store Scene Response 13) Get Scene Membership Response
ZCL_ON_OFF	Enable the following commands: 1) On 2) Off 3) Toggle
ZCL_LEVEL_CTRL	Enable the following commands: 1) Move to Level 2) Move 3) Step 4) Stop

	<ul style="list-style-type: none"> 5) Move to Level with On/Off 6) Move with On/Off 7) Step with On/Off 8) Stop with On/Off
ZCL_ALARMS	<p>Enable the following commands:</p> <ul style="list-style-type: none"> 1) Reset Alarm 2) Reset All Alarms 3) Get Alarm 4) Reset Alarm Log 5) Alarm 6) Get Alarm Response
ZCL_LOCATION	<p>Enable the following commands:</p> <ul style="list-style-type: none"> 1) Set Absolute Location 2) Set Device Configuration 3) Get Device Configuration 4) Get Location Data 5) Device Configuration Response 6) Location Data Response 7) Location Data Notification 8) Compact Location Data Notification 9) RSSI Ping
ZCL_ZONE	<p>Enable the following commands:</p> <ul style="list-style-type: none"> 1) Zone Status Change Notification 2) Zone Enroll Request 3) Zone Enroll Response
ZCL_ACE	<p>Enable the following commands:</p> <ul style="list-style-type: none"> 1) Arm 2) Bypass 3) Emergency 4) Fire 5) Panic 6) Get Zone ID Map 7) Get Zone Information 8) Arm Response 9) Get Zone ID Map Response 10) Get Zone Information Response
ZCL_WD	<p>Enable the following commands:</p> <ul style="list-style-type: none"> 1) Start Warning 2) Squawk