Z-Stack Lighting
Application Programming Interface

Document Number: SWRA418

Texas Instruments, Inc.
San Diego, California USA

Version	Description	Date
1.0	Initial release	12/26/2012
1.1	Added Control-Bridge related functionality	11/26/2013

TABLE OF CONTENTS

LIST OF FIGURES

1. Introduction

1.1 Purpose

The purpose of this document is to define the Z-Stack Lighting API. This API allows the User Application to access the ZigBee Light Link profile functionality.

1.2 Scope

This document enumerates the APIs provided by the Z-Stack Lighting's ZigBee Light Link profile. This document does not provide details of other profiles and functional domains. Furthermore, this document doesn't explain the ZigBee Light Link concepts.

1.3 Definitions, Abbreviations and Acronyms

Term	Definition
MT_APP	Monitor and Test interface for Application layer
OSAL	Operating System Abstraction Layer
Touch-link	A method of finding and pairing devices nearby, based on received signal strength. Its operation is composed of device discovery and network settings transfer, and is defined in the ZLL specification.
ZCL	ZigBee Cluster Library
ZLL	ZigBee Light Link

1.4 Applicable Documents

1. ZigBee document 075123r03ZB, ZigBee Cluster Library Specification
2. ZigBee document 11-0037, ZigBee Light Link Profile Specification
3. Texas Instruments document SWRA197, Z-Stack ZCL API
4. Texas Instruments document SWRA198, Z-Stack Monitor and Test API

1.5 API Overview

The Z-Stack Lighting software provides APIs to the User application layer to:

1. Operate as a ZLL Lighting device.
2. Operate as a ZLL Controller device.
3. Generate and register for receiving ZLL commissioning utility cluster Request and Response commands.
4. Generate and register for receiving other ZCL commands, including enhancements defined in the ZLL specification.
5. Send general ZCL commands and initiate ZLL functionalities from remote processor using MT interface.

1.6 ZLL Profile Task Creation

The ZLL profile uses a dedicated task to handle the ZLL touch-link commissioning and the network adjustments required by the ZLL specification. This task is required to be initiated before any application using it.

A Lighting device should use the Target module. Its task should be initiated by OSAL by calling `zllTarget_Init()` with its task ID as argument. Events will be directed to it by assigning `zllTarget_event_loop` in OSAL's event handlers array.

A Controller device should use the Initiator module. Its task should be initiated by OSAL by calling `zllInitiator_Init()` with its task ID as argument. Events will be directed to it by assigning `zllInitiator_event_loop` in OSAL's event handlers array.

Similar functionality should be implemented by the user application for its endpoints.

1.7 Application/Profile Registration

1.7.1 Basic Registration

The ZLL application uses two faces to identify and publish its endpoint's capabilities on the network:

a. Simple descriptor, used in the context of ZigBee service discovery in ZigBee networks.
b. Device information record, used in the context of ZLL touch-link.

Those two data structures must be prepared and referenced as arguments in the registration functions:

```
ZStatus_t zllTarget_RegisterApp( SimpleDescriptionFormat_t *simpleDesc,
                                 zclLLDeviceInfo_t *pDeviceInfo );
ZStatus_t zllInitiator_RegisterApp( SimpleDescriptionFormat_t *simpleDesc,
                                    zclLLDeviceInfo_t *pDeviceInfo );
```

used by Lighting and Controller devices, respectfully.
The simple descriptor provided will be registered with Z-Stack's Application Framework, while the device information will be added to the ZLL sub-devices list.

1.7.2 System Messages Registration

A controller application may register for unprocessed system messages, by calling:

```
ZStatus_t zllInitiator_RegisterForMsg( uint8 taskId );
```

This feature is useful when using the MT_APP interface for remote processing, allowing the application to receive data from a host processor, required for generating ZCL commands.

2. Lighting Device (Target) API

2.1 Introduction

The following interface allow a Lighting application to initiate new ZLL processes by direct function calls, and to participate with initiated ZLL processes by callback function registration.

2.2 Target Start Device

2.2.1 Description

Start the Lighting device in the network if it's not Factory New. Otherwise, determine the network parameters and wait for a touch-link command.

2.2.2 Prototype

```
ZStatus_t zllTarget_InitDevice( void );
```

2.2.3 Parameter Details

None.

2.2.4 Return

`ZStatus_t` - status definitions found in ZComDef.h.

2.3 Target Reset To Factory New

2.3.1 Description

Reset the device to Factory New state.

2.3.2 Prototype

```
void zllTarget_ResetToFactoryNew( void );
```

2.3.3 Parameter Details

None.

2.3.4 Return

None.

2.4 Target Permit Join

2.4.1 Description

Set the router's permit join flag, to allow or deny classical commissioning of other ZigBee devices via this device.

2.4.2 Prototype

```
ZStatus_t zllTarget_PermitJoin( uint8 duration );
```

2.4.3 Parameter Details

`duration` – positive number of seconds up to aplcMaxPermitJoinDuration seconds to enable, 0 to disable

2.4.4 Return

`ZStatus_t` – status definitions found in ZComDef.h.

2.5 Target Start Classical Commissioning

2.5.1 Description

Start Classical Commissioning.

2.5.2 Prototype

```
ZStatus_t zllTarget_ClassicalCommissioningStart( void );
```

2.5.3 Parameter Details

None.

2.5.4 Return

`ZStatus_t` - status definitions found in ZComDef.h.

2.6 Target Register Identify Callback Function

2.6.1 Description

Register an Application's callback function, to be called when Identify Request command is received.

2.6.2 Prototype

```
void zllTarget_RegisterIdentifyCB( zclGCB_Identify_t pfnIdentify );
```

2.6.3 Parameter Details

`pfnIdentify` – pointer to a ZCL general Identify callback function. The callback is described in [3].

2.6.4 Return

None.

3. Controller Device (Initiator) API

3.1 Introduction

The following interface allows a Controller application to initiate new ZLL processes by direct function calls, and to participate with initiated ZLL processes by callback function registration.

3.2 Initiator Start Device

3.2.1 Description

Start the Controller device in the network if it's not Factory New. Otherwise, determine the network parameters and wait for a touch-link command.

3.2.2 Prototype

```
ZStatus_t zllTarget_InitDevice( void );
```

3.2.3 Parameter Details

None.

3.2.4 Return

`ZStatus_t` - status definitions found in ZComDef.h.

3.3 Initiator Start Touch-link Commissioning

3.3.1 Description

Start device discovery, scanning for other devices in the vicinity of the originator (initiating first part of the touch-link process). Device discovery shall only be initiated by address assignment capable devices. To perform device discovery, the initiator shall broadcast inter-PAN Scan Requests, spaced at an interval of `ZLL_APLC_SCAN_TIME_BASE_DURATION` seconds.

3.3.2 Prototype

```
ZStatus_t zllInitiator_StartDevDisc( void );
```

3.3.3 Parameter Details

None.

3.3.4 Return

`ZStatus_t` - status definitions found in ZComDef.h.

3.4 Initiator Abort Touch-link Commissioning

3.4.1 Description

Abort touch-link device discovery. Successful execution could be done before Network Start/Join commands are sent. Until then, since no device parameters such as network settings are altered, the touch-link is still reversible.

3.4.2 Prototype

```
ZStatus_t zllInitiator_AbortTL( void );
```

3.4.3 Parameter Details

None.

3.4.4 Return
`ZStatus_t` - status definitions found in ZComDef.h.

3.5 Initiator Change Channel

3.5.1 Description

Instigate channel change mechanism according to ZLL Frequency Agility.

3.5.2 Prototype

```
ZStatus_t zllInitiator_ChannelChange( uint8 targetChannel );
```

3.5.3 Parameter Details

`targetChannel` – target channel to move to, or next channel if 0.

3.5.4 Return
`ZStatus_t` - status definitions found in ZComDef.h.

3.6 Initiator Reset to Factory New

3.6.1 Description

Reset the device to Factory New state.

3.6.2 Prototype

```
void zllInitiator_ResetToFactoryNew( void );
```

3.6.3 Parameter Details

None.

3.6.4 Return
None.

3.7 Initiator Start Classical Commissioning

3.7.1 Description

Start Classical Commissioning.

3.7.2 Prototype

```
ZStatus_t zllInitiator_ClassicalCommissioningStart( void );
```

3.7.3 Parameter Details

None.

3.7.4 Return
`ZStatus_t` - status definitions found in ZComDef.h.

3.8 Initiator Send Reset to Factory New

3.8.1 Description

Send Reset to Factory New Request command to the selected target of the current touch-link transaction.
Note - this function should be called within no later than `ZLL_APLC_INTER_PAN_TRANS_ID_LIFETIME` ms from the Scan Request initiated by the call described at section 3.3.

3.8.2 Prototype

```
ZStatus_t zllInitiator_ResetToFNSelectedTarget( void );
```

3.8.3 Parameter Details

None.

3.8.4 Return

`ZStatus_t` - status definitions found in ZComDef.h.

3.9 Initiator Rejoin If Orphaned

3.9.1 Description

Try to Rejoin network if the device is orphaned.

It is recommended to be called upon existing from deep sleep mode.

3.9.2 Prototype

```
ZStatus_t zllInitiator_OrphanedRejoin( void );
```

3.9.3 Parameter Details

None.

3.9.4 Return

`ZStatus_t` - status definitions found in ZComDef.h.

3.10 Initiator Send Endpoint Info

3.10.1 Description

Send EP Info commands. This is a wrapper function for the explicit call described in section 4.5.
Note that an application calling this function may not have the information about other sub-devices registered on the device, which is required for the command payload.

3.10.2 Prototype

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

3.10.3 Parameter Details

`srcEP` – source endpoint.

`dstAddr` – destination address.

`seqNum` – ZCL sequence number.

3.10.4 Return

`ZStatus_t` - status definitions found in ZComDef.h.

3.11 Initiator Register Identify Callback Function

3.11.1 Description

Register an Application's callback function, to be called when Identify Request command is received.

3.11.2 Prototype

```
void zllInitiator_RegisterIdentifyCB( zclGCB_Identify_t pfnIdentify );
```

3.11.3 Parameter Details

`pfnIdentify` – pointer to a ZCL general Identify callback function. The callback is described in [3].

3.11.4 Return

None.

3.12 Initiator Register Notify Touch-link Callback Function

3.12.1 Description

Register an Application's callback function, to be called upon a successful touch-link.

3.12.2 Prototype

```
void zllInitiator_RegisterNotifyTLCB( zll_NotifyAppTLCB_t pfnNotifyApp );
```

3.12.3 Parameter Details

`pfnNotifyApp` – pointer to a touch-link notification callback function is defined as follows:

```
typedef ZStatus_t (*zll_NotifyAppTLCB_t)( epInfoRec_t *pData );
```

where `epInfoRec_t` is endpoint information record defined in section 7.1.2.3.8 of [2]. See section 4.7.3 for the structure description.

3.12.4 Return

None.

3.13 Initiator Register Reset to FN Callback Function

3.13.1 Description

Register an Application's callback function, to be called when Reset to Factory New is processed.

3.13.2 Prototype

```
void zllInitiator_RegisterResetAppCB( zclGCB_BasicReset_t pfnResetApp );
```

3.13.3 Parameter Details

`pfnResetApp` – pointer to ZCL general callback function for processing Reset to Factory Defaults command, described in [3].

3.13.4 Return

None.

3.14 Initiator Register Target Selection Callback Function

3.14.1 Description

Register an application's target selection callback function, to select a target from the discovered devices during a touch-link scan.
If no selection callback function is registered, default target selection will be done according to strongest RSSI.

3.14.2 Prototype

```
void zllInitiator_RegisterSelectDiscDevCB( zll_SelectDiscDevCB_t
                                                        pfnSelectDiscDev );
```

3.14.3 Parameter Details

`pfnSelectDiscDev` - pointer to the selection callback function, defined as follows:

```
typedef uint8 (*zll_SelectDiscDevCB_t)
              ( const zclLLScanRsp_t *newScanRsp, int8 newRssi );
```

This callback is called for each device which responded to scan, allows deciding whether to select it as the chosen target, according to its Scan Response and its RSSI measurement.
The selection function should return `TRUE` when the device referred by the passed arguments should be marked as the new selected target for the current discovery session; or `FALSE` otherwise.
Note `newScanRsp` information should be copied if used beyond the call scope.

3.14.4 Return

None.

3.15 Initiator Bridge Start Network

3.15.1 Description

Start a new ZLL network, without coordinator or Trust Center.
This function is available only for devices built as ZigBee Routers (e.g. control-bridge).

3.15.2 Prototype

```
ZStatus_t zllInitiator_BridgeStartNetwork( void );
```

3.15.3 Parameter Details

none.

3.15.4 Return

`ZStatus_t` – status definitions found in ZComDef.h.

3.16 Initiator Permit Join

3.16.1 Description

Set the router permit join flag, to allow or deny classical commissioning by other ZigBee devices.
This function is available only for devices built as ZigBee Routers.

3.16.2 Prototype

```
ZStatus_t zllInitiator_PermitJoin( uint8 duration );
```

3.16.3 Parameter Details

`duration` – positive number of seconds up to aplcMaxPermitJoinDuration seconds to enable, 0 to disable

3.16.4 Return

`ZStatus_t` – status definitions found in ZComDef.h.

4. ZLL Commissioning Cluster (Utility) Commands Set

4.1 Overview

The ZLL Commissioning Cluster provides an API to the User application layer to generate Request and Response commands and to register application's command callback functions.

This section describes only the ZCL part relevant for this cluster's utility commands, which are instigated by the application layer. The rest of this cluster's commands are touch-link inter-PAN commands, which are controlled internally by the ZLL profile layer, and are not part of the application/profile API.

4.2 Application/Profile Registration

The ZCL Foundation provides APIs to the Application/Profile to register their Attribute List, Cluster Option List, Attribute Data Validation, and Cluster Library Handler callback functions.
The `zclLL_RegisterCmdCallbacks()` function is used to register the Application's Command callback functions with the ZLL Commissioning Cluster. The command callback input parameter is of type:

```
typedef struct
{

  // Received Server Commands

  zclLL_GetGrpIDsReqCB_t   pfnGetGrpIDsReq;
  zclLL_GetEPListReqCB_t   pfnGetEPListReq;

  // Received Client Commands

  zclLL_EndpointInfoCB_t   pfnEndpointInfo;
  zclLL_GetGrpIDsRspCB_t   pfnGetGrpIDsRsp;
  zclLL_GetEPListRspCB_t   pfnGetEPListRsp;

} zclLL_AppCallbacks_t;
```

If the application does not support some of the callbacks, the table entry shall be input as `NULL`.

4.3 Send Get Group Identifiers Request Command

4.3.1 Description

This function is used to send out a Get Group Identifiers Request command.

4.3.2 Prototype

```
ZStatus_t zclLL_Send_GetGrpIDsReq( uint8 srcEP,
                                   afAddrType_t *dstAddr,
                                   zclLLGetGrpIDsReq_t *pReq,
                                   uint8 disableDefaultRsp,
                                   uint8 seqNum );
```

4.3.3 Parameter Details

`srcEP` - sending application's endpoint

`dstAddr` – target destination address

`pReq` - request parameters. The structure of the request is as follows:

```
typedef struct
{
```



```
        uint8 startIndex; // Start index
      } zclLLGetGrpIDsReq_t;
```

`disableDefaultRsp` - whether to disable the Default Response command

`seqNum` – ZCL sequence number

4.3.4 Return

`ZStatus_t` - status definitions found in ZComDef.h.

4.4 Send Get Endpoint List Request Command

4.4.1 Description

This function is used to send out a Get Endpoint List Request command.

4.4.2 Prototype

```
ZStatus_t zclLL_Send_GetEPListReq( uint8 srcEP,
                                   afAddrType_t *dstAddr,
                                   zclLLGetEPListReq_t *pReq,
                                   uint8 disableDefaultRsp,
                                   uint8 seqNum );
```

4.4.3 Parameter Details

`srcEP` - sending application's endpoint

`dstAddr` - target destination address

`pReq` - request parameters. The request structure is as follows:

```
      typedef struct
      {
        uint8 startIndex; // Start index
      } zclLLGetEPListReq_t;
```

`disableDefaultRsp` - whether to disable the Default Response command

`seqNum` – ZCL sequence number

4.4.4 Return

`ZStatus_t` - status definitions found in ZComDef.h.

4.5 Send Endpoint Information Command

4.5.1 Description

This function is used to send out an Endpoint Information command.

4.5.2 Prototype

```
ZStatus_t zclLL_Send_EndpointInfo( uint8 srcEP,
                                   afAddrType_t *dstAddr,
                                   zclLLEndpointInfo_t *pCmd,
                                   uint8 disableDefaultRsp,
                                   uint8 seqNum );
```


4.5.3 Parameter Details

srcEP - sending application's endpoint

dstAddr - target destination address

pCmd – command parameters. The command structure is as follows:

```
typedef struct
{
  uint8 endpoint;                    // Endpoint identifier
  uint16 profileID;                  // Profile identifier
  uint16 deviceID;                   // Device identifier
  uint8 version;                     // Version
  uint8 grpIdCnt;                    // Group identifier count
} zclLLDeviceInfo_t;
```

disableDefaultRsp - whether to disable the Default Response command

seqNum – ZCL sequence number

4.5.4 Return

ZStatus_t - status definitions found in ZComDef.h.

4.6 Send Get Group Identifiers Response Command

4.6.1 Description

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

4.6.2 Prototype

```
ZStatus_t zclLL_Send_GetGrpIDsRsp( uint8 srcEP,
                                   afAddrType_t *dstAddr,
                                   zclLLGetGrpIDsRsp_t *pRsp,
                                   uint8 disableDefaultRsp,
                                   uint8 seqNum );
```

4.6.3 Parameter Details

srcEP - sending application's endpoint

dstAddr - target destination address

pRsp - response parameters. The response structure is as follows:

```
typedef struct
{
  uint8 total;        // total number of group ids supported by device
  uint8 startIndex;   // Start index
  uint8 cnt;          // Number of entries in the group info record
  grpInfoRec_t *grpInfoRec; // Group information records
} zclLLGetGrpIDsRsp_t;

typedef struct
{
  uint16 grpID;  // Group identifier
  uint8 grpType; // Group type
} grpInfoRec_t;
```

disableDefaultRsp - whether to disable the Default Response command

seqNum – ZCL sequence number.

4.6.4 Return

`ZStatus_t` - status definitions found in ZComDef.h.

4.7 Send Get Endpoint List Response Command

4.7.1 Description

This function is used to send out a Get Endpoint List Response command.

4.7.2 Prototype

```
ZStatus_t zclLL_Send_GetEPListRsp( uint8 srcEP,
                                   afAddrType_t *dstAddr,
                                   zclLLGetEPListRsp_t *pRsp,
                                   uint8 disableDefaultRsp,
                                   uint8 seqNum );
```

4.7.3 Parameter Details

`srcEP` - sending application's endpoint

`dstAddr` - target destination address

`pRsp` - response parameters. The response structure is as follows:

```
typedef struct
{
  uint8 total;         // total number of endpoints supported by device
  uint8 startIndex;    // Start index
  uint8 cnt;           // Number of entries in the endpoint info record
  epInfoRec_t *epInfoRec; // Endpoint information records
} zclLLGetEPListRsp_t;

typedef struct
{
  uint16 nwkAddr;   // Network address
  uint8 endpoint;   // Endpoint identifier
  uint16 profileID; // Profile identifier
  uint16 deviceID;  // Device identifier
  uint8 version;    // Version
} epInfoRec_t;
```

`disableDefaultRsp` - whether to disable the Default Response command

`seqNum` – ZCL sequence number.

4.7.4 Return

`ZStatus_t` - status definitions found in ZComDef.h.

5. ZLL Enhancements to the General ZigBee Cluster Library

5.1 Overview

The ZLL enhancements provide APIs to the User application layer to send and register for receiving Request and Response ZCL commands defined in the ZLL specification. The enhancements to the ZCL layer are enabled when the flag `ZCL_LIGHT_LINK_ENHANCE` is set on compilation time.
For full details about ZCL commands used by ZLL applications but not defined in the ZLL specification, please refer to [3].

5.2 Application/Profile Registration

The ZLL enhancements are integral part of the API provided by the General functional domain and Lighting functional domain to register the Application's Command callback functions.
The `zclGeneral_RegisterCmdCallbacks()` API is used to register the Application's Command callback functions with the general ZCL. The ZLL enhancements are marked in yellow:

```
typedef struct
{
  zclGCB_BasicReset_t               pfnBasicReset;
  zclGCB_Identify_t                 pfnIdentify;
#ifdef ZCL_EZMODE
  zclGCB_IdentifyEZModeInvoke_t     pfnIdentifyEZModeInvoke;
  zclGCB_IdentifyUpdateCommState_t  pfnIdentifyUpdateCommState;
#endif
  zclGCB_IdentifyTriggerEffect_t    pfnIdentifyTriggerEffect;
  zclGCB_IdentifyQueryRsp_t         pfnIdentifyQueryRsp;
  zclGCB_OnOff_t                    pfnOnOff;
  zclGCB_OnOff_OffWithEffect_t      pfnOnOff_OffWithEffect;
  zclGCB_OnOff_OnWithRecallGlobalScene_t  pfnOnOff_OnWithRecallGlobalScene;
  zclGCB_OnOff_OnWithTimedOff_t     pfnOnOff_OnWithTimedOff;
#ifdef ZCL_LEVEL_CTRL
  zclGCB_LevelControlMoveToLevel_t  pfnLevelControlMoveToLevel;
  zclGCB_LevelControlMove_t         pfnLevelControlMove;
  zclGCB_LevelControlStep_t         pfnLevelControlStep;
  zclGCB_LevelControlStop_t         pfnLevelControlStop;
#endif
#ifdef ZCL_GROUPS
  zclGCB_GroupRsp_t                 pfnGroupRsp;
#endif
#ifdef ZCL_SCENES
  zclGCB_SceneStoreReq_t            pfnSceneStoreReq;
  zclGCB_SceneRecallReq_t           pfnSceneRecallReq;
  zclGCB_SceneRsp_t                 pfnSceneRsp;
#endif
#if ZCL_ALARMS
  zclGCB_Alarm_t                    pfnAlarm;
#endif
#ifdef SE_UK_EXT
  zclGCB_GetEventLog_t              pfnGetEventLog;
  zclGCB_PublishEventLog_t          pfnPublishEventLog;
#endif
  zclGCB_Location_t                 pfnLocation;
```



```
  zclGCB_LocationRsp_t              pfnLocationRsp;
} zclGeneral_AppCallbacks_t;
```

If the application does not support some of the callbacks, the table entry shall be input as NULL.
The `zclLighting_RegisterCmdCallbacks()` API is used to register the Application's Command callback functions with the Lighting ZCL. The ZLL enhancements are marked in yellow:

```
typedef struct
{
  zclLighting_ColorControl_MoveToHue_t            pfnColorControl_MoveToHue;
  zclLighting_ColorControl_MoveHue_t              pfnColorControl_MoveHue;
  zclLighting_ColorControl_StepHue_t              pfnColorControl_StepHue;
  zclLighting_ColorControl_MoveToSaturation_t
                                          pfnColorControl_MoveToSaturation;
  zclLighting_ColorControl_MoveSaturation_t   pfnColorControl_MoveSaturation;
  zclLighting_ColorControl_StepSaturation_t   pfnColorControl_StepSaturation;
  zclLighting_ColorControl_MoveToHueAndSaturation_t
                                      pfnColorControl_MoveToHueAndSaturation;
  zclLighting_ColorControl_MoveToColor_t         pfnColorControl_MoveToColor;
  zclLighting_ColorControl_MoveColor_t           pfnColorControl_MoveColor;
  zclLighting_ColorControl_StepColor_t           pfnColorControl_StepColor;
  zclLighting_ColorControl_MoveToColorTemperature_t
                                      pfnColorControl_MoveToColorTemperature;
  zclLighting_ColorControl_EnhancedMoveToHue_t
                                        pfnColorControl_EnhancedMoveToHue;
  zclLighting_ColorControl_EnhancedMoveHue_t pfnColorControl_EnhancedMoveHue;
  zclLighting_ColorControl_EnhancedStepHue_t pfnColorControl_EnhancedStepHue;
  zclLighting_ColorControl_EnhancedMoveToHueAndSaturation_t
                             pfnColorControl_EnhancedMoveToHueAndSaturation;
  zclLighting_ColorControl_ColorLoopSet_t       pfnColorControl_ColorLoopSet;
  zclLighting_ColorControl_StopMoveStep_t       pfnColorControl_StopMoveStep;
} zclLighting_AppCallbacks_t;
```

5.3 Send Trigger Effect

5.3.1 Description

Call to send out a Trigger Effect command.

5.3.2 Prototype

```
ZStatus_t zclGeneral_SendIdentifyTriggerEffect( uint8 srcEP,
                                                afAddrType_t *dstAddr,
                                                uint8 effectId,
                                                uint8 effectVariant,
                                                uint8 disableDefaultRsp,
                                                uint8 seqNum );
```

5.3.3 Parameter Details

`srcEP` - Sending application's endpoint

`dstAddr` – target destination address.

`effectId` - identify effect to use

`effectVariant` - which variant of effect to be triggered

`disableDefaultRsp` - whether to disable the Default Response command

`seqNum` - ZCL sequence number

5.3.4 Return

`ZStatus_t` - status definitions found in ZComDef.h.

5.4 Send Enhanced Add Scene command

5.4.1 Description

Send an Enhanced Add Scene command.

5.4.2 Prototype

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

5.4.3 Parameter Details

`srcEP` - Sending application's endpoint

`dstAddr` – target destination address.

`scene` – pointer to the scene structure.

`disableDefaultRsp` - whether to disable the Default Response command

`seqNum` - ZCL sequence number.

5.4.4 Return

`ZStatus_t` - status definitions found in ZComDef.h.

5.5 Send Enhanced View Scene command

5.5.1 Description

Send an Enhanced View Scene command.

5.5.2 Prototype

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

5.5.3 Parameter Details

`srcEP` - Sending application's endpoint

`dstAddr` – target destination address.

`groupID` – Group identification

`sceneID` – Scene identification

`disableDefaultRsp` - whether to disable the Default Response command

`seqNum` - ZCL sequence number.

5.5.4 Return

`ZStatus_t` - status definitions found in ZComDef.h.

5.6 Send Enhanced Add Scene Response command

5.6.1 Description

Send an Enhanced Add Scene Response command.

5.6.2 Prototype

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

5.6.3 Parameter Details

`srcEP` - Sending application's endpoint

`dstAddr` – target destination address.

`groupID` – Group identification

`sceneID` – Scene identification

`disableDefaultRsp` - whether to disable the Default Response command

`seqNum` - ZCL sequence number.

5.6.4 Return

`ZStatus_t` - status definitions found in ZComDef.h.

5.7 Send Enhanced View Scene Response command

5.7.1 Description

Send an Enhanced View Scene Response command.

5.7.2 Prototype

```
ZStatus_t zclGeneral_SendSceneEnhancedViewResponse( uint16 srcEP,
                                                    afAddrType_t *dstAddr,
                                                    uint8 status,
                                                   zclGeneral_Scene_t *scene,
                                                    uint8 disableDefaultRsp,
                                                    uint8 seqNum );
```

5.7.3 Parameter Details

`srcEP` - Sending application's endpoint

`dstAddr` – target destination address.

`status` – scene command status.

`scene` – pointer to the scene structure.

`disableDefaultRsp` - whether to disable the Default Response command

`seqNum` - ZCL sequence number.

5.7.4 Return

`ZStatus_t` - status definitions found in ZComDef.h.

5.8 Send Copy Scene command

5.8.1 Description

Send a Copy Scene command.

5.8.2 Prototype

```
ZStatus_t zclGeneral_SendSceneCopy( uint8 srcEP,
                                    afAddrType_t *dstAddr,
                                    uint8 mode,
                                    uint16 groupIDFrom,
                                    uint8 sceneIDFrom,
                                    uint16 groupIDTo,
                                    uint8 sceneIDTo,
                                    uint8 disableDefaultRsp,
                                    uint8 seqNum );
```

5.8.3 Parameter Details

`srcEP` - Sending application's endpoint

`dstAddr` – target destination address.

`mode` – how scene copy is to proceed.

`groupIDFrom` – group from which the scene is to be copied.

`sceneIDFrom` – scene from which the scene is to be copied.

`groupIDTo` – group to which the scene is to be copied.

`sceneIDTo` – scene to which the scene is to be copied.

`disableDefaultRsp` - whether to disable the Default Response command

`seqNum` - ZCL sequence number.

5.8.4 Return

`ZStatus_t` - status definitions found in ZComDef.h.

5.9 Send Copy Scene Response command

5.9.1 Description

Send a Copy Scene Response command.

5.9.2 Prototype

```
ZStatus_t zclGeneral_SendSceneCopyResponse( uint8 srcEP,
                                            afAddrType_t *dstAddr,
                                            uint8 status,
                                            uint16 groupIDFrom,
                                            uint8 sceneIDFrom,
                                            uint8 disableDefaultRsp,
                                            uint8 seqNum );
```

5.9.3 Parameter Details

`srcEP` - Sending application's endpoint

`dstAddr` – target destination address.

`status` – status of copy scene attempt

`groupIDFrom` – group from which the scene was copied.

`sceneIDFrom` – scene from which the scene was copied.

`disableDefaultRsp` - whether to disable the Default Response command

`seqNum` - ZCL sequence number.

5.9.4 Return

`ZStatus_t` - status definitions found in ZComDef.h.

5.10 Send Off with Effect command

5.10.1 Description

Call to send out an Off with Effect Command.

5.10.2 Prototype

```
ZStatus_t zclGeneral_SendOnOff_CmdOffWithEffect( uint8 srcEP,
                                                 afAddrType_t *dstAddr,
                                                 uint8 effectId,
                                                 uint8 effectVariant,
                                                 uint8 disableDefaultRsp,
                                                 uint8 seqNum );
```

5.10.3 Parameter Details

`srcEP` - Sending application's endpoint

`dstAddr` – target destination address.

`effectId` - fading effect to use when switching light off.

`effectVariant` - which variant of effect to be triggered

`disableDefaultRsp` - whether to disable the Default Response command

`seqNum` - ZCL sequence number.

5.10.4 Return

`ZStatus_t` - status definitions found in ZComDef.h.

5.11 Send On with Recall Global Scene command

5.11.1 Description

Call to send out an On with Recall Global Scene command.

5.11.2 Prototype

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

5.11.3 Parameter Details

`srcEP` - Sending application's endpoint

`dstAddr` – target destination address.

`disableDefaultRsp` - whether to disable the Default Response command

`seqNum` - ZCL sequence number.

5.11.4 Return

`ZStatus_t` - status definitions found in ZComDef.h.

5.12 Send On with Timed Off command

5.12.1 Description

Call to send out an On with Timed Off Command.

5.12.2 Prototype

```
ZStatus_t zclGeneral_SendOnOff_CmdOnWithTimedOff( uint8 srcEP,
                                                  afAddrType_t *dstAddr,
                                                  zclOnOffCtrl_t onOffCtrl,
                                                  uint16 onTime,
                                                  uint16 offWaitTime,
                                                  uint8 disableDefaultRsp,
                                                  uint8 seqNum );
```

5.12.3 Parameter Details

`srcEP` - Sending application's endpoint

`dstAddr` – target destination address.

`onOffCtrl` - how the lamp is to be operated.

`onTime` - the length of time (in $1/10^{th}$s second) that the lamp is to remain on, before automatically turning off.

`offWaitTime` - the length of time (in $1/10^{th}$s second) that the lamp shall remain off, and guarded to prevent an on command turning the light back on.

`disableDefaultRsp` - whether to disable the Default Response command

`seqNum` - ZCL sequence number.

5.12.4 Return

`ZStatus_t` - status definitions found in ZComDef.h.

5.13 Send Enhanced Move to Hue command

5.13.1 Description

Call to send out an Enhanced Move To Hue Command

5.13.2 Prototype

```
ZStatus_t zclLighting_ColorControl_Send_EnhancedMoveToHueCmd( uint8 srcEP,
                                                              afAddrType_t *dstAddr,
                                                              uint16 enhancedHue,
                                                              uint8 direction,
                                                              uint16 transitionTime,
                                                              uint8 disableDefaultRsp,
                                                              uint8 seqNum );
```

5.13.3 Parameter Details

`srcEP` - Sending application's endpoint

`dstAddr` – target destination address.

`enhancedHue` - a target extended hue for lamp.

`direction` - direction of hue change. Could be either:
`LIGHTING_MOVE_TO_HUE_DIRECTION_SHORTEST_DISTANCE`,
`LIGHTING_MOVE_TO_HUE_DIRECTION_LONGEST_DISTANCE`,
`LIGHTING_MOVE_TO_HUE_DIRECTION_UP` or `LIGHTING_MOVE_TO_HUE_DIRECTION_DOWN`.

`transitionTime` - time to perform the color change, equal of the value of the field in 1/10 seconds.

`disableDefaultRsp` - whether to disable the Default Response command

`seqNum` - ZCL sequence number.

5.13.4 Return

`ZStatus_t` - status definitions found in ZComDef.h.

5.14 Send Enhanced Move Hue command

5.14.1 Description

Call to send out an Enhanced Move Hue Command.

5.14.2 Prototype

```
ZStatus_t zclLighting_ColorControl_Send_EnhancedMoveHueCmd( uint8 srcEP,
                                                            afAddrType_t *dstAddr,
                                                            uint8 moveMode,
                                                            uint16 rate,
                                                            uint8 disableDefaultRsp,
                                                            uint8 seqNum );
```

5.14.3 Parameter Details

`srcEP` - Sending application's endpoint

`dstAddr` – target destination address.

`moveMode` - `LIGHTING_MOVE_HUE_STOP`, `LIGHTING_MOVE_HUE_UP` or `LIGHTING_MOVE_HUE_DOWN`.

`rate` - the movement in steps per second, where step is a change in the device's hue of one unit.

`disableDefaultRsp` - whether to disable the Default Response command

`seqNum` - ZCL sequence number.

5.14.4 Return

`ZStatus_t` - status definitions found in ZComDef.h.

5.15 Send Enhanced Step Hue command

5.15.1 Description

Call to send out an Enhanced Step Hue Command.

5.15.2 Prototype

```
ZStatus_t zclLighting_ColorControl_Send_EnhancedStepHueCmd( uint8 srcEP,
                                                            afAddrType_t *dstAddr,
                                                            uint8 stepMode,
                                                            uint16 stepSize,
                                                            uint16 transitionTime,
                                                            uint8 disableDefaultRsp,
                                                            uint8 seqNum );
```

5.15.3 Parameter Details

`srcEP` - Sending application's endpoint

`dstAddr` – target destination address.

`stepMode` - `LIGHTING_STEP_HUE_UP` or `LIGHTING_STEP_HUE_DOWN`.

`stepSize` - change to the current value of the device's hue.

`transitionTime` – the time that shall be taken to perform the step, in units of $1/10^{th}$ of a second.

`disableDefaultRsp` - whether to disable the Default Response command

`seqNum` - ZCL sequence number.

5.15.4 Return

`ZStatus_t` - status definitions found in ZComDef.h.

5.16 Send Enhanced Move to Hue and Saturation command

5.16.1 Description

Call to send out an Enhanced Move to Hue and Saturation command.

5.16.2 Prototype

```
ZStatus_t zclLighting_ColorControl_Send_EnhancedMoveToHueAndSaturationCmd(
                                                            uint8 srcEP,
                                                            afAddrType_t *dstAddr,
                                                            uint16 enhancedHue,
                                                            uint8 saturation,
                                                            uint16 transitionTime,
                                                            uint8 disableDefaultRsp,
                                                            uint8 seqNum );
```

5.16.3 Parameter Details

srcEP - Sending application's endpoint

dstAddr – target destination address.

enhancedHue - a target Enhanced hue for lamp.

saturation - a target saturation.

transitionTime – time to move, equal of the value of the field in 1/10th seconds.

disableDefaultRsp - whether to disable the Default Response command

seqNum - ZCL sequence number.

5.16.4 Return

ZStatus_t - status definitions found in ZComDef.h.

5.17 Send Color Loop Set command

5.17.1 Description

Call to send out a Color Loop Set command.

5.17.2 Prototype

```
ZStatus_t zclLighting_ColorControl_Send_ColorLoopSetCmd( uint8 srcEP,
                                afAddrType_t *dstAddr,
                                zclCCColorLoopSet_updateFlags_t updateFlags,
                                uint8 action,
                                uint8 direction,
                                uint16 time,
                                uint16 startHue,
                                uint8 disableDefaultRsp,
                                uint8 seqNum);
```

5.17.3 Parameter Details

srcEP - Sending application's endpoint

dstAddr – target destination address.

updateFlags - which color loop attributes to update before the color loop is started. The structure is defined as follows:

```
typedef union
{
  zclCCColorLoopSet_updateFlagsBits_t bits;
  uint8 byte;
} zclCCColorLoopSet_updateFlags_t;

typedef struct
{
  unsigned int action:1;
  unsigned int direction:1;
  unsigned int time:1;
  unsigned int startHue:1;
  unsigned int reserved:4;
} zclCCColorLoopSet_updateFlagsBits_t;
```

action - action to take for the color loop.

`direction` – direction for the color loop (decrement or increment).

`time` – number of seconds over which to perform a full color loop.

`startHue` – starting hue to use for the color loop.

`disableDefaultRsp` - whether to disable the Default Response command

`seqNum` - ZCL sequence number.

5.17.4 Return

`ZStatus_t` - status definitions found in ZComDef.h.

5.18 Send Stop Move Step command

5.18.1 Description

Send out a Stop Move Step Command

5.18.2 Prototype

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

5.18.3 Parameter Details

`srcEP` - Sending application's endpoint

`dstAddr` – target destination address.

`disableDefaultRsp` - whether to disable the Default Response command

`seqNum` - ZCL sequence number.

5.18.4 Return

`ZStatus_t` - status definitions found in ZComDef.h.

6. ZLL MT Interface

6.1 Overview

The Z-Stack Lighting allows to initiate ZLL remote procedure calls on all devices and to transfer ZCL commands to and from controller devices, via serial port connection to the device, using MT_APP interface.
This interface also allows returning indication for a successful touch-link, using a new extension to MT_APP.
This feature allows flexibility in the product design, allowing the Z-Stack processor component to be part of host processor architecture.
These features require the Z-Stack Lighting project to be compiled with the MT_TASK and MT_APP_FUNC compilation flags.
See document [4] for more details.

6.2 ZLL Remote Procedure Calls

It is possible to initiate ZLL procedures remotely by using the *APP_MSG* command with the *ClusterId* attribute set to `ZLL_MT_APP_RPC_CLUSTER_ID`.

ZCL header's command ID	Processed by Controller device	Processed by Lighting device	Supported payload
`ZLL_MT_APP_RPC_CMD_TOUCHLINK`	Yes	No	None
`ZLL_MT_APP_RPC_CMD_RESET_TO_FN`	Yes	Yes	None
`ZLL_MT_APP_RPC_CMD_CH_CHANNEL`	Yes	No	Target channel (1 byte)
`ZLL_MT_APP_RPC_CMD_JOIN_HA`	Yes	Yes	None
`ZLL_MT_APP_RPC_CMD_PERMIT_JOIN`	Router Only	Yes	Duration in sec (1 byte)
`ZLL_MT_APP_RPC_CMD_SEND_RESET_TO_FN`	Yes	No	None
`ZLL_MT_APP_RPC_CMD_START_DISTRIB_NWK`	Router Only	No	None

Table 1: Remote Procedure Calls used with MT_APP

6.2.1 Touch Link RPC

By receiving ZLL MT_APP RPC with the command ID of the ZCL header set to the value of `ZLL_MT_APP_RPC_CMD_TOUCHLINK` (0x01) a controller device will start a touch-link sequence.

6.2.2 Reset to Factory New RPC

By receiving ZLL MT_APP RPC with the command ID of the ZCL header set to the value of `ZLL_MT_APP_RPC_CMD_RESET_TO_FN` (0x02) a controller device or a lighting device will reset itself to FN state.

6.2.3 Channel Change RPC

By receiving ZLL MT_APP RPC with the command ID of the ZCL header set to the value of `ZLL_MT_APP_RPC_CMD_CH_CHANNEL` (0x03) a controller device will send Mgmt_NWK_Update_req command, according to the Frequency agility defined in the ZLL specification, with the target channel that was set in the RPC payload.

6.2.4 Classical Commissioning RPC

By receiving ZLL MT_APP RPC with the command ID of the ZCL header set to the value of ZLL_MT_APP_RPC_CMD_JOIN_HA (0x04) a controller device or a lighting device will start classical commissioning to scan and join available ZigBee network.

6.2.5 Permit Join RPC

By receiving ZLL MT_APP RPC with the command ID of the ZCL header set to the value of ZLL_MT_APP_RPC_CMD_PERMIT_JOIN (0x05) a router device will set its permit join flag to TRUE for one minute.

6.2.6 Send Reset to Factory New RPC

By receiving ZLL MT_APP RPC with the command ID of the ZCL header set to the value of ZLL_MT_APP_RPC_CMD_SEND_RESET_TO_FN (0x06) a controller device will send the Reset to Factory New Request command to the most recent Touch Link target, if the transaction ID is still valid.

6.2.7 Start Distributed Network RPC

By receiving ZLL MT_APP RPC with the command ID of the ZCL header set to the value of ZLL_MT_APP_RPC_CMD_START_DISTRIB_NWK (0x07) a router controller device will start an ad-hoc ZLL distributed security network, without a preliminary interaction with another device.

6.3 ZCL Commands Remote Generator

In Z-Stack Lighting controller project, an application could generate general ZCL commands (e.g. Color Loop Set command or Read Attribute command), according to instructions sent from remote or host device over MT_APP interface.

In order to initiate the sending of ZCL command, the *APP_MSG* command should be used by a host device, as defined in section 3.3.1.1 of [4], where the *AppEndpoint* attribute is the application generating the ZCL command (SAMPLEREMOTE_ENDPOINT in the case of the provided Sample Remote application).

Note that the first byte of the *Message* attribute is used to determine the addressing mode, according to the following key: 0x01 = groupcast, 0x02 = 16-bit short address, 0x0F = broadcast.

Following that first byte of the *Message* attribute are the ZCL header and payload bytes, according to [1].

It is required for the controller application to implement a handler function for MT_SYS_APP_MSG system message in its task events processing function, and to register for receiving such messages (see 1.7.2).

6.4 ZCL Commands Remote Receiver

6.4.1 Description

In a Z-Stack Lighting controller project, the following function could be used to send received ZCL commands by an application (e.g. Read Response command), over MT_APP to a remote or host device.
Messages will be sent in Z-Tool format as ***APP_MSG_RSP*.**

SRSP:

Byte: 1	1	1	1	2	1	2	1	0-128
Length = 0x08-0x87	Cmd0 = 0x69	Cmd1 = 0x80	AppEndpoint	SrcAddress	SrcEndpoint	ClusterID	MsgLen	Message

6.4.2 Prototype


```
void MT_ZllSendZCLCmd( uint8 appEP, uint16 shortAddr, uint8 endpoint,
                       uint16 clusterID, zclFrameHdr_t *pZclHdr,
                       uint8 payloadLen, uint8 *pPayload );
```

6.4.3 Parameter Details

`appEP` - The user application endpoint (original message's destination endpoint).

`shortAddr` – source network address.

`endpoint` - source endpoint.

`clusterID` - ZCL cluster ID.

`pZclHdr` - pointer to the received ZCL header.

`payloadLen` - length of payload.

`pPayload` - pointer to the payload buffer.

6.4.4 Return

None.

6.5 MT_APP ZLL Touch Link Indication

In a Z-Stack Lighting controller project, a notification to the application could be instigated upon a successful touch-link (see section 3.12). When MT_APP is enabled, an indication with the same payload (endpoint information record entry, according to section 7.1.2.3.8 and Figure 55 in [2]) will be transmitted to the host device.

AREQ:

Byte: 1	1	1	2	1	2	2	1
Length = 0x08	Cmd0 = 0x49	Cmd1 = 0x81	Network Address	Endpoint Identifier	Profile Identifier	Device Identifier	Version

