Z-Stack Commissioning Cluster Application Programming Interface

Document Number: SWRA363

Texas Instruments, Inc.
San Diego, California USA

Version	Description	Date
1.0	Initial release	05/04/2011

TABLE OF CONTENTS

1. INTRODUCTION 1
1.1 PURPOSE 1
1.2 SCOPE 1
1.3 DEFINITIONS, ABBREVIATIONS AND ACRONYMS 1
1.4 APPLICABLE DOCUMENTS 1
2. API OVERVIEW 2
2.1 OVERVIEW 2
2.2 APPLICATION/PROFILE REGISTRATION 2
2.3 APPLICATION CREATION 2
3. COMMISSIONING CLUSTER 3
3.1 INTRODUCTION 3
3.2 SEND RESTART DEVICE COMMAND 4
3.3 SEND SAVE STARTUP PARAMETERS COMMAND 4
3.4 SEND RESTORE STARTUP PARAMETERS COMMAND 5
3.5 SEND RESET STARTUP PARAMETERS COMMAND 6
3.6 SEND RESTART DEVICE RESPONSE 6
3.7 SEND SAVE STARTUP PARAMETERS RESPONSE 7
3.8 SEND RESTORE STARTUP PARAMETERS RESPONSE 7
3.9 SEND RESET STARTUP PARAMETERS RESPONSE 8
3.10 REGISTER COMMAND CALLBACKS 9
3.11 RESTART DEVICE COMMAND CALLBACK 9
3.12 SAVE STARTUP PARAMETERS COMMAND CALLBACK 10
3.13 RESTORE STARTUP PARAMETERS COMMAND CALLBACK 10
3.14 RESET STARTUP PARAMETERS COMMAND CALLBACK 11
3.15 RESTART DEVICE RESPONSE CALLBACK 11
3.16 SAVE STARTUP PARAMETERS RESPONSE CALLBACK 12
3.17 RESTORE STARTUP PARAMETERS RESPONSE CALLBACK 13
3.18 RESET STARTUP PARAMETERS RESPONSE CALLBACK 13

LIST OF FIGURES

FIGURE 1: SCREEN SHOT OF UPDATED AND ADDED ZCL FILES TO SUPPORT COMMISSIONING CLUSTER 3

1. Introduction

1.1 Purpose

The purpose of this document is to define the Commissioning Cluster API. This API allows the User Application to access the Commissioning Cluster functionality.

1.2 Scope

This document enumerates the APIs provided by the Commissioning Cluster. This document does not provide details of other clusters and functional domains. Furthermore, this document doesn't explain the Commissioning Cluster concepts.

1.3 Definitions, Abbreviations and Acronyms

Term	Definition
CBA	Commercial Building Automation (now called ZBA)
CC	Commissioning Cluster
Client	A cluster interface which is listed in the output cluster list of the simple descriptor on an endpoint. Typically this interface sends commands to 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.
ZBA	ZigBee Building Automation (same as CBA)
ZCL	ZigBee Cluster Library

1.4 Applicable Documents

1. ZigBee document 075123r03ZB, ZigBee Cluster Library Specification
2. ZigBee document 053516r11ZB, Commercial Building Automation Specification
3. Texas Instruments document SWRA197, Z-Stack ZCL API

2. API Overview

2.1 Overview

The Commissioning Cluster provides APIs to the User application layer to:

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

2.2 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 General functional domain provides an API to register the Application's Command callback functions.

The `zclCC_RegisterCmdCallbacks()` API is used to register the Application's Command callback functions with the Commissioning Cluster. The command callback input parameter is of type:

```
typedef struct
{
  zclCC_Restart_Device_t            pfnRestart_Device;
  zclCC_Save_StartupParams_t        pfnSave_StartupParams;
  zclCC_Restore_StartupParams_t     pfnRestore_StartupParams;
  zclCC_Reset_StartupParams_t       pfnReset_StartupParams;
  zclCC_Restart_DeviceRsp_t         pfnRestart_DeviceRsp;
  zclCC_Save_StartupParamsRsp_t     pfnSave_StartupParamsRsp;
  zclCC_Restore_StartupParamsRsp_t  pfnRestore_StartupParamsRsp;
  zclCC_Reset_StartupParamsRsp_t    pfnReset_StartupParamsRsp;
} zclCC_AppCallbacks_t;
```

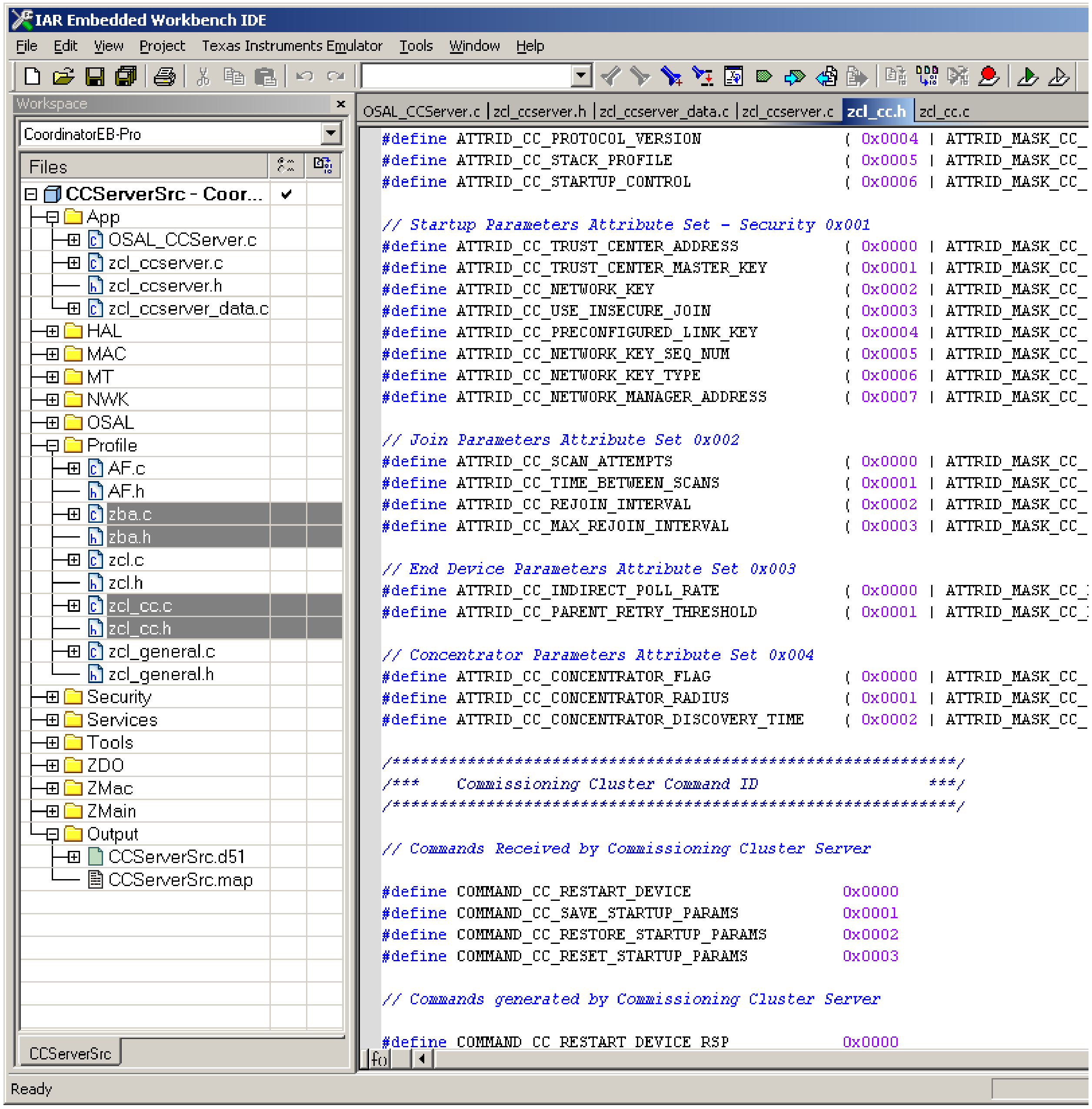
The prototype for each command callback function is defined in section 6 of this document. If the application does not support some of the callbacks, the table entry shall be input as NULL.

2.3 Application Creation

Section 2.5 in [3] outlines the steps to be taken when creating a new ZCL application. Instead of the last step explained in [3] Section 2.5.3, the application's initialization function `zcl<AppName>_Init()` should register its *simple descriptor* with the ZBA profile by calling `zba_Init()`, defined in the *zba.c* module. The application also needs to call `zclCC_RegisterCmdCallbacks()`, defined in the *zcl_cc.c* module, to register the application's command callback functions.

To support the ZBA profile, the user also needs to add a number ZCL source files which can be found here:

- $PROJ_DIR$\..\..\..\..\..\Components\stack\zcl
- $PROJ_DIR$\..\Source



```
#define ATTRID_CC_PROTOCOL_VERSION              ( 0x0004 | ATTRID_MASK_CC_
#define ATTRID_CC_STACK_PROFILE                 ( 0x0005 | ATTRID_MASK_CC_
#define ATTRID_CC_STARTUP_CONTROL               ( 0x0006 | ATTRID_MASK_CC_

// Startup Parameters Attribute Set - Security 0x001
#define ATTRID_CC_TRUST_CENTER_ADDRESS          ( 0x0000 | ATTRID_MASK_CC_
#define ATTRID_CC_TRUST_CENTER_MASTER_KEY       ( 0x0001 | ATTRID_MASK_CC_
#define ATTRID_CC_NETWORK_KEY                   ( 0x0002 | ATTRID_MASK_CC_
#define ATTRID_CC_USE_INSECURE_JOIN             ( 0x0003 | ATTRID_MASK_CC_
#define ATTRID_CC_PRECONFIGURED_LINK_KEY        ( 0x0004 | ATTRID_MASK_CC_
#define ATTRID_CC_NETWORK_KEY_SEQ_NUM           ( 0x0005 | ATTRID_MASK_CC_
#define ATTRID_CC_NETWORK_KEY_TYPE              ( 0x0006 | ATTRID_MASK_CC_
#define ATTRID_CC_NETWORK_MANAGER_ADDRESS       ( 0x0007 | ATTRID_MASK_CC_

// Join Parameters Attribute Set 0x002
#define ATTRID_CC_SCAN_ATTEMPTS                 ( 0x0000 | ATTRID_MASK_CC_
#define ATTRID_CC_TIME_BETWEEN_SCANS            ( 0x0001 | ATTRID_MASK_CC_
#define ATTRID_CC_REJOIN_INTERVAL               ( 0x0002 | ATTRID_MASK_CC_
#define ATTRID_CC_MAX_REJOIN_INTERVAL           ( 0x0003 | ATTRID_MASK_CC_

// End Device Parameters Attribute Set 0x003
#define ATTRID_CC_INDIRECT_POLL_RATE            ( 0x0000 | ATTRID_MASK_CC_
#define ATTRID_CC_PARENT_RETRY_THRESHOLD        ( 0x0001 | ATTRID_MASK_CC_

// Concentrator Parameters Attribute Set 0x004
#define ATTRID_CC_CONCENTRATOR_FLAG             ( 0x0000 | ATTRID_MASK_CC_
#define ATTRID_CC_CONCENTRATOR_RADIUS           ( 0x0001 | ATTRID_MASK_CC_
#define ATTRID_CC_CONCENTRATOR_DISCOVERY_TIME   ( 0x0002 | ATTRID_MASK_CC_

/*************************************************************/
/***     Commissioning Cluster Command ID                  ***/
/*************************************************************/

// Commands Received by Commissioning Cluster Server

#define COMMAND_CC_RESTART_DEVICE              0x0000
#define COMMAND_CC_SAVE_STARTUP_PARAMS         0x0001
#define COMMAND_CC_RESTORE_STARTUP_PARAMS      0x0002
#define COMMAND_CC_RESET_STARTUP_PARAMS        0x0003

// Commands generated by Commissioning Cluster Server

#define COMMAND_CC_RESTART_DEVICE_RSP          0x0000
```

Figure 1: Screen shot of updated and added ZCL files to support Commissioning Cluster

3. Commissioning Cluster

3.1 Introduction

The Commissioning Cluster provides the following commands and corresponding responses:

- Restart Device
- Save Startup Parameters
- Restore Startup Parameters
- Reset Startup Parameters

- Restart Device Response
- Save Startup Parameters Response
- Restore Startup Parameters Response
- Reset Startup Parameters Response

The Commissioning Cluster consists of a group of attributes and commands. Detailed attribute list and command frame format are listed in [1] section 3.15. The CC commands and responses are all implemented in *zcl_cc.c* and *zcl_cc.h* files.

3.2 Send Restart Device Command

3.2.1 Description

This function is used to send out Restart Device Command to the Commissioning Cluster Server.

3.2.2 Prototype

```
ZStatus_t zclCC_Send_RestartDevice( uint8 srcEP,
                                    afAddrType_t *dstAddr,
                                    zclCCRestartDevice_t *pCmd,
                                    uint8 disableDefaultRsp,
                                    uint8 seqNum )
```

3.2.3 Parameter Details

`srcEP` - Sending application's endpoint

`dstAddr` - Where you want the message to go

`pCmd` – Pointer to Restart Device command data structure defined as follows:

```
      typedef struct
      {
        uint8 options;
        uint8 delay;
        uint8 jitter;
      } zclCCRestartDevice_t;
```

`disableDefaultRsp` - Decides default response is necessary or not

`seqNum` - sequence number of the command packet

3.2.4 Return

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

3.3 Send Save Startup Parameters Command

3.3.1 Description

This function is used to send out Save Startup Parameters Command to the Commissioning Cluster Server.

3.3.2 Prototype

```
ZStatus_t zclCC_Send_SaveStartupParams( uint8 srcEP,
                                         afAddrType_t *dstAddr,
                                         zclCCStartupParams_t *pCmd,
                                         uint8 disableDefaultRsp,
                                         uint8 seqNum )
```

3.3.3 Parameter Details

`srcEP` - Sending application's endpoint

`dstAddr` - Where you want the message to go

`pCmd` – Pointer to Startup Parameters command data structure defined as follows:

```
      typedef struct
      {
        uint8 options;
        uint8 index;
      } zclCCStartupParams_t;
```

`disableDefaultRsp` - Decides default response is necessary or not

`seqNum` - sequence number of the command packet

3.3.4 Return

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

3.4 Send Restore Startup Parameters Command

3.4.1 Description

This function is used to send out Restore Startup Parameters Command to the Commissioning Cluster Server.

3.4.2 Prototype

```
ZStatus_t zclCC_Send_RestoreStartupParams( uint8 srcEP,
                                           afAddrType_t *dstAddr,
                                           zclCCStartupParams_t *pCmd,
                                           uint8 disableDefaultRsp,
                                           uint8 seqNum )
```

3.4.3 Parameter Details

`srcEP` - Sending application's endpoint

`dstAddr` - Where you want the message to go

`pCmd` – Pointer to Startup Parameters command data structure defined as follows:

```
      typedef struct
      {
        uint8 options;
        uint8 index;
      } zclCCStartupParams_t;
```

`disableDefaultRsp` - Decides default response is necessary or not

`seqNum` - sequence number of the command packet

3.4.4 Return

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

3.5 Send Reset Startup Parameters Command

3.5.1 Description

This function is used to send out Reset Startup Parameters Command to the Commissioning Cluster Server.

3.5.2 Prototype

```
ZStatus_t zclCC_Send_ResetStartupParams( uint8 srcEP,
                                         afAddrType_t *dstAddr,
                                         zclCCStartupParams_t *pCmd,
                                         uint8 disableDefaultRsp,
                                         uint8 seqNum )
```

3.5.3 Parameter Details

`srcEP` - Sending application's endpoint

`dstAddr` - Where you want the message to go

`pCmd` – Pointer to Startup Parameters command data structure defined as follows:

```
      typedef struct
      {
        uint8 options;
        uint8 index;
      } zclCCStartupParams_t;
```

`disableDefaultRsp` - Decides default response is necessary or not

`seqNum` - sequence number of the command packet

3.5.4 Return

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

3.6 Send Restart Device Response

3.6.1 Description

This function is used to send Restart Device Response to the Commissioning Cluster Client.

3.6.2 Prototype

```
ZStatus_t zclCC_Send_RestartDeviceRsp( uint8 srcEP,
                                       afAddrType_t *dstAddr,
                                       zclCCServerParamsRsp_t *pRsp,
                                       uint8 disableDefaultRsp,
                                       uint8 seqNum )
```

3.6.3 Parameter Details

`srcEP` - Sending application's endpoint

`dstAddr` - Where you want the message to go

`pRsp` – Pointer to Server Parameters Response data structure defined as follows:

```
      typedef struct
      {
        uint8 status;
      } zclCCServerParamsRsp_t;
```

`disableDefaultRsp` - Decides default response is necessary or not

`seqNum` - sequence number of the command packet

3.6.4 Return

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

3.7 Send Save Startup Parameters Response

3.7.1 Description

This function is used to send Save Startup Parameters Response to the Commissioning Cluster Client.

3.7.2 Prototype

```
ZStatus_t zclCC_Send_ SaveStartupParamsRsp( uint8 srcEP,
                                            afAddrType_t *dstAddr,
                                            zclCCServerParamsRsp_t *pRsp,
                                            uint8 disableDefaultRsp,
                                            uint8 seqNum )
```

3.7.3 Parameter Details

`srcEP` - Sending application's endpoint

`dstAddr` - Where you want the message to go

`pRsp` – Pointer to Server Parameters Response data structure defined as follows:

```
      typedef struct
      {
        uint8 status;
      } zclCCServerParamsRsp_t;
```

`disableDefaultRsp` - Decides default response is necessary or not

`seqNum` - sequence number of the command packet

3.7.4 Return

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

3.8 Send Restore Startup Parameters Response

3.8.1 Description

This function is used to send Restore Startup Parameters Response to the Commissioning Cluster Client.

3.8.2 Prototype

```
ZStatus_t zclCC_Send_ RestoreStartupParamsRsp( uint8 srcEP,
                                               afAddrType_t *dstAddr,
                                                            zclCCServerParamsRsp_t *pRsp,
                                                            uint8 disableDefaultRsp,
                                                            uint8 seqNum )
```

3.8.3 Parameter Details

`srcEP` - Sending application's endpoint

`dstAddr` - Where you want the message to go

`pRsp` – Pointer to Server Response Response data structure defined as follows:

```
      typedef struct
      {
        uint8 status;
      } zclCCServerParamsRsp_t;
```

`disableDefaultRsp` - Decides default response is necessary or not

`seqNum` - sequence number of the command packet

3.8.4 Return

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

3.9 Send Reset Startup Parameters Response

3.9.1 Description

This function is used to send Reset Startup Parameters Response to the Commissioning Cluster Client.

3.9.2 Prototype

```
ZStatus_t zclCC_Send_ResetDeviceRsp( uint8 srcEP,
                                     afAddrType_t *dstAddr,
                                     zclCCServerParamsRsp_t *pRsp,
                                     uint8 disableDefaultRsp,
                                     uint8 seqNum )
```

3.9.3 Parameter Details

`srcEP` - Sending application's endpoint

`dstAddr` - Where you want the message to go

`pRsp` – Pointer to Server Parameters Response data structure defined as follows:

```
      typedef struct
      {
        uint8 status;
      } zclCCServerParamsRsp_t;
```

`disableDefaultRsp` - Decides default response is necessary or not

`seqNum` - sequence number of the command packet

3.9.4 Return

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

3.10 Register Command Callbacks

3.10.1 Description

This callback is called to register an application's command callbacks.

3.10.2 Prototype

```
ZStaus_t zclCC_RegisterCmdCallbacks( uint8 endpoint,
                                     zclCC_AppCallbacks_t *callbacks );
```

3.10.3 Parameter Details

`srcEP` – Application's endpoint.

`callbacks -` pointer to the callback record defined in Section 2.2

3.10.4 Return

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

3.11 Restart Device Command Callback

3.11.1 Description

This callback is called to process an incoming Restart Device Command. On receipt of this command, the device responds with Restart Device Response.

3.11.2 Prototype

```
typedef void (*zclCC_Restart_Device_t)( zclCCRestartDevice_t*pCmd,
                                        afAddrType_t *srcAddr,
                                        uint8 seqNum );
```

3.11.3 Parameter Details

`pCmd` – Received command. The structure of the command is as follows:

```
      typedef struct
      {
        uint8 options;
        uint8 index;
      } zclCCStartupParams_t;
```

`srcAddr` – Requestor's address.

`seqNum` – ZCL sequence number.

3.11.4 Return

None.

3.12 Save Startup Parameters Command Callback

3.12.1 Description

This callback is called to process an incoming Save Startup Parameters Command. On receipt of this command, the device responds with ‘Save Startup Parameters’ Response.

3.12.2 Prototype

```
typedef void (*zclCC_ Save_StartupParams_t)( zclCCStartupParams_t*pCmd,
                                            afAddrType_t *srcAddr,
                                            uint8 seqNum );
```

3.12.3 Parameter Details

`pCmd` – Received command. The structure of the command is as follows:

```
      typedef struct
      {
        uint8 options;
        uint8 index;
      } zclCCStartupParams_t;
```

`srcAddr` – Requestor’s address.

`seqNum` – ZCL sequence number.

3.12.4 Return

None.

3.13 Restore Startup Parameters Command Callback

3.13.1 Description

This callback is called to process an incoming Restore Startup Parameters Command. On receipt of this command, the device responds with ‘Restore Startup Parameters’ Response.

3.13.2 Prototype

```
typedef void (*zclCC_ Restore_StartupParams_t)( zclCCStartupParams_t*pCmd,
                                                afAddrType_t *srcAddr,
                                                uint8 seqNum );
```

3.13.3 Parameter Details

`pCmd` – Received command. The structure of the command is as follows:

```
      typedef struct
      {
        uint8 options;
        uint8 index;
      } zclCCStartupParams_t;
```

`srcAddr` – Requestor's address.

`seqNum` – ZCL sequence number.

3.13.4 Return

None.

3.14 Reset Startup Parameters Command Callback

3.14.1 Description

This callback is called to process an incoming Reset Startup Parameters Command. On receipt of this command, the device responds with 'Reset Startup Parameters' Response.

3.14.2 Prototype

```
typedef void (*zclCC_ Reset_StartupParams_t)( zclCCStartupParams_t*pCmd,
                                             afAddrType_t *srcAddr,
                                             uint8 seqNum );
```

3.14.3 Parameter Details

`pCmd` – Received command. The structure of the command is as follows:

```
      typedef struct
      {
        uint8 options;
        uint8 index;
      } zclCCStartupParams_t;
```

`srcAddr` – Requestor's address.

`seqNum` – ZCL sequence number.

3.14.4 Return

None.

3.15 Restart Device Response Callback

3.15.1 Description

This callback is called to process an incoming Restart Device Response.

3.15.2 Prototype

```
typedef void (*zclCC_Restart_DeviceRsp_t)( zclCCServerParamsRsp_t*pRsp,
                                           afAddrType_t *srcAddr,
                                           uint8 seqNum );
```

3.15.3 Parameter Details

`pRsp` – Received response. The structure of the command is as follows:

```
      typedef struct
      {
        uint8 status;
      } zclCCServerParamsRsp_t;
```

`srcAddr` – Requestor's address.

`seqNum` – ZCL sequence number.

3.15.4 Return

None.

3.16 Save Startup Parameters Response Callback

3.16.1 Description

This callback is called to process an incoming Save Startup Parameters Response.

3.16.2 Prototype

```
typedef void (*zclCC_Save_StartupParamsRsp_t)( zclCCServerParamsRsp_t*pRsp,
                                               afAddrType_t *srcAddr,
                                               uint8 seqNum );
```

3.16.3 Parameter Details

`pRsp` – Received response. The structure of the command is as follows:

```
      typedef struct
      {
        uint8 status;
      } zclCCServerParamsRsp_t;
```

`srcAddr` – Requestor's address.

`seqNum` – ZCL sequence number.

3.16.4 Return

None.

3.17 Restore Startup Parameters Response Callback

3.17.1 Description

This callback is called to process an incoming Restore Startup Parameters Response.

3.17.2 Prototype

```
typedef void (*zclCC_Restore_StartupParamsRsp_t)(zclCCServerParamsRsp_t*pRsp,
                                                  afAddrType_t *srcAddr,
                                                  uint8 seqNum );
```

3.17.3 Parameter Details

`pRsp` – Received response. The structure of the command is as follows:

```
      typedef struct
      {
        uint8 status;
      } zclCCServerParamsRsp_t;
```

`srcAddr` – Requestor's address.

`seqNum` – ZCL sequence number.

3.17.4 Return

None.

3.18 Reset Startup Parameters Response Callback

3.18.1 Description

This callback is called to process an incoming Reset Startup Parameters Response.

3.18.2 Prototype

```
typedef void (*zclCC_Reset_StartupParamsRsp_t)( zclCCServerParamsRsp_t*pRsp,
                                                 afAddrType_t *srcAddr,
                                                 uint8 seqNum );
```

3.18.3 Parameter Details

`pRsp` – Received response. The structure of the command is as follows:

```
      typedef struct
      {
        uint8 status;
      } zclCCServerParamsRsp_t;
```

`srcAddr` – Requestor’s address.

`seqNum` – ZCL sequence number.

3.18.4 Return

None.