RT58x Zigbee SDK User Guide V1.0

Table of Contents

1. Introduction

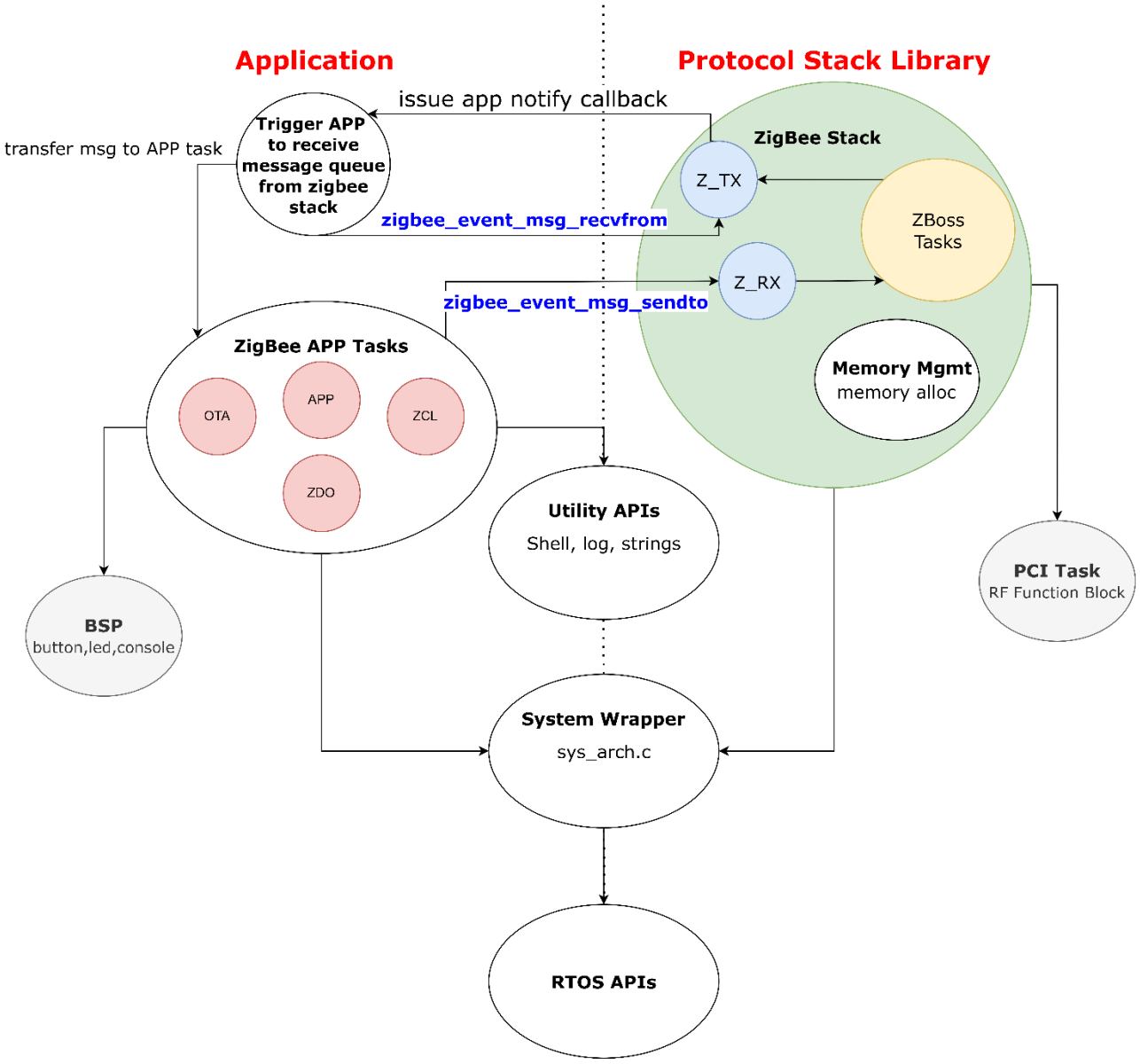
This document describes the interface to the Rafael Zigbee Stack Library. The upper layer tasks use provided APIs to communicate with Rafael Zigbee Stack Library through a series of primitives as defined in following sections.

The Rafael Zigbee Stack Library is easy to integrate with Rafael RT58x series SDK.

2. System architecture

This section provides an overview of the zigbee stack library operation and layered system architecture.

2.1 Task architecture



2.1.1 Zigbee Stack

Rafael ‘s Zigbee stack sub-system.

2.1.2 PCI Task

Rafael’s RF function block controller.

2.1.3 System wrapper

OS wrapper for real time OS.

2.1.4 Utility APIs

Useful functions and general macro definitions.

Ex.: debug log, queue, list, FSM.

2.1.5 Zigbee APP Tasks

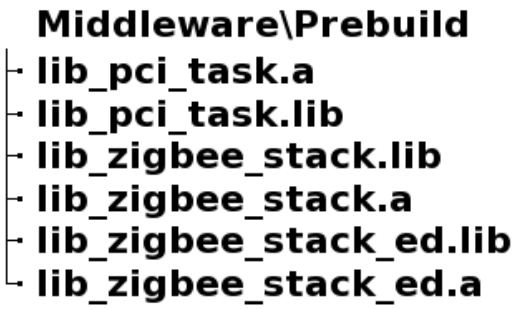
Handle BSP, Appliction, ZCL and product behavior.

2.1.6 RTOS

Zigbee SDK is based on FreeRTOS.

3. Zigbee SDK introduction

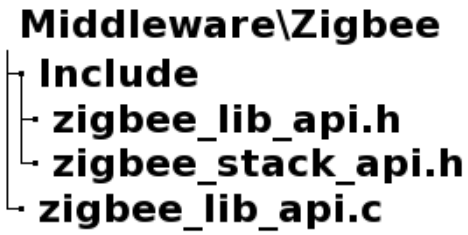
3.1 Prebuild Library



Zigbee project must include lib_pci_task, zigbee_stack.lib

Zigbee_stack_ed.lib is used to an end-device for sleeping.

3.2 Zigbee Stack Library



Include:Stack APIs

Zigbee_lib_api.c : Useful zigbee APIs.

Ex: zigbee_nwk_start, zigbee_nwk_join, zigbee_zcl_data_request

3.3 Zigbee Example

Project\Application\Zigbee_Demo

- **Gateway**
- **Light**
- **OTA**
- **Switch**

3.3.1 Gateway

This application is a Zigbee Coordinator device for a simple gateway.

Simple Gateway application includes following ZCL clusters:

- Identify (s/c)
- Basic (s)
- On/Off (c)
- Scenes (c)
- Groups (c)
- Level control(c)
- Color control(c)
- OTA Upgrade (s)

3.3.2 Light

This application is a Zigbee Router device for a light device.

Light application includes following ZCL clusters:

- Identify (s)
- Basic (s)
- On/Off (s)
- Scenes (s)
- Groups (s)
- Level control(s)

- Color control(s)

3.3.3 Switch

This application is a Zigbee End-device for a switch device.

Light application includes following ZCL clusters:

- Identify (s)
- Basic (s)
- On/Off (c)
- Level control(c)
- Color control(c)

3.3.4 OTA

This application is a Zigbee Router device for a light device.

Light application includes following ZCL clusters:

- Basic (s)
- OTA Upgrade (c)

4. Stack APIs and Primitive

This section provides and overview the APIs

4.1 Initial

Issued Zigbee Stack initialization function and application layer initialization.

```
int
zigbee_stack_init(zigbee_cfg_t *pt_cfg);
```

- **Parameter**

Name	Type	Valid Range	Description
pt _cfg	Pointer	-	Pointer of the application configuration.

- **Return value**

Name	Type	Valid Range	Description
-	Integer	0 ~ -1	The status of initialization. 0:Success, -1:Fail

4.2 Event message receive

Receive TLV packet from Zigbee Stack Library.
When Zigbee Stack Library call the event indication of register callback function, application layer can use this function to receive event message from Zigbee Stack Library.

```
int
zigbee_event_msg_recvfrom(uint8_t *pu8_buf,
                          uint32_t *pu32_buf_length);
```

- **Parameter**

Name	Type	Valid Range	Description
pu8_buf	Pointer	-	Pointer of the message buffer.
pu32_buf_length	Pointer	-	Pointer of the message length.

- **Return value**

Name	Type	Valid Range	Description
-	Integer	0 ~ -1	The status of received result. 0:Success, -1:Fail

4.3 Event message sendto

Send message to Zigbee Stack Library.
This function will call memory allocator of register function, and copy message buffer, then send to Zigbee Stack Library.

```
int
zigbee_event_msg_sendto(sys_tlv_t *pt_tlv);
```

- **Parameter**

Name	Type	Valid Range	Description
pt_tlv	Pointer	-	Pointer of the message TLV buffer.

- **Return value**

Name	Type	Valid Range	Description
-	Integer	0 ~ -1	The status of sent result. 0:Success, -1:Fail

4.4 Data structure

4.4.1 Application configuration

Configuration of application layer for Zigbee Stack initial.

```
typedef struct ZIGBEE_CONFIG_T
{
        zb_af_device_ctx_t  *p_zigbee_device_contex_t;
        pf_evt_indication   *pf_evt_indication;
} zigbee_cfg_t;
```

- **Parameter**

Name	Type	Valid Range	Description
p_zigbee_device_contex_t	Pointer	-	Pointer of the zigbee device context.
pf_evt_indication	Pointer		Pointer of the event message indication.

4.4.2 Message TLV

```
typedef struct SYSTEM_TLV_FORMAT
{
        uint16_t    type;
        uint16_t    length;
        uint8_t     value[];
} sys_tlv_t;
```

➢ **Parameter**

Name	Type	Valid Range	Description
type	Integer	0 – 0x999	Payload identifier.
length	Integer	0 – 0xFFFF	The length of the payload data.
value	Various	-	Payload buffer.

4.5 Message Primitives

Following primitives are supported:

Name	Identifier	Value
Network start request	ZIGBEE_EVT_TYPE_NWK_START_REQ	0x0000
Join request	ZIGBEE_EVT_TYPE_NWK_JOIN_REQ	0x0001
ZCL data request	ZIGBEE_EVT_TYPE_ZCL_DATA_REQ	0x0002
Simple description request	ZIGBEE_EVT_TYPE_ZDO_SIMPLE_DESC_REQ	0x0003
Bind request	ZIGBEE_EVT_TYPE_ZDO_BIND_REQ	0x0004
Active endpoint request	ZIGBEE_EVT_TYPE_ZDO_ACT_EP_REQ	0x0005
Network reset request	ZIGBEE_EVT_TYPE_NWK_RESET_REQ	0x0006
Rejoin request	ZIGBEE_EVT_TYPE_NWK_REJOIN_REQ	0x0007
Permit join request	ZIGBEE_EVT_TYPE_NWK_PERMIT_JOIN_REQ	0x0008
OTA file insert request	ZIGBEE_EVT_TYPE_OTA_FILE_INSERT_REQ	0x0009
Network start indication	ZIGBEE_EVT_TYPE_ZDO_START_IDC	0x1000
Device announce indication	ZIGBEE_EVT_TYPE_DEVICE_ANNCE_IDC	0x1001
~~Device associated indication~~	~~ZIGBEE_EVT_TYPE_DEVICE_ASSOCIATED_IDC~~	~~0x1002~~
~~Leave indication~~	~~ZIGBEE_EVT_TYPE_LEAVE_INDICATION_IDC~~	~~0x1003~~
~~Rejoin failure indication~~	~~ZIGBEE_EVT_TYPE_REJOIN_FAILURE_IDC~~	~~0x1004~~
~~Pan ID conflict indication~~	~~ZIGBEE_EVT_TYPE_PANID_CONFLICT_IDC~~	~~0x1005~~
Active endpoint indication	ZIGBEE_EVT_TYPE_ZDO_ACT_EP_IDC	0x1006
Simple description indication	ZIGBEE_EVT_TYPE_ZDO_SIMPLE_DESC_IDC	0x1007
ZCL data indication	ZIGBEE_EVT_TYPE_ZCL_DATA_IDC	0x1008

4.5.1 Network start request

This primitive allows application layer to start a network and device configuration.
The semantics of this primitive are as flow:

- **Request**

```
typedef struct ZIGBEE_NWK_START_REQ_T
{
    uint32_t deviceRole     : 2;
    uint32_t maxChild       : 8;
    uint32_t                : 5;
    // end device only
    uint32_t rx_always_on   : 1;
    uint32_t keepalive      : 16; // micro second

    uint32_t channelMask;
    uint16_t PANID;

    uint8_t  ieeeAddr[8];
}zigbee_nwk_start_req_t;
```

Name	Type	Valid Range	Description
deviceRole	Integer	0-2	0: Coordinator 1: Router 2: End-Device
maxChild	Integer	0-255	The maximum child.
rx_always_on	Bool	0-1	End-Device only RX allways on(none-sleep)
keepalive	Integer	0-65535	Keep alive timeout
channelMask	Integer		Channel mask 16 = 1 < 16
PANID	Integer	0-0xFFFE	PAN ID
ieeeAddr	Various	-	IEEE Address

4.5.2 Join request

This primitive allows application layer to join a network.

The semantics of this primitive are as flow:

- **Request**

  No parameter.

4.5.3 ZCL data request

This primitive allow application layer to send ZCL data.

The semantics of this primitive are as flow:

- **Request**

```
typedef struct ZIGBEE_ZCL_DATA_REQ_T
{
    uint32_t dstAddr        : 16;
    uint32_t dstEndpint     : 8;
    uint32_t srcEndPoint    : 8;

    uint32_t clusterID             : 16;
    uint32_t cmd                   : 8;
    uint32_t specific              : 1;
    uint32_t disableDefaultRsp     : 1;
    uint32_t direction             : 1;
    uint32_t addrmode              : 3;
    uint32_t                       : 2;

    uint32_t manuCode          : 16;
    uint32_t cmdFormatLen      : 16;

    uint8_t cmdFormat[];

}zigbee_zcl_data_req_t;
```

Name	Type	Valid Range	Description
dstAddr	Integer	0 – 0xFFFF	Destination address
dstEndpoint	Integer	0 – 0xFF	Destination endpoint
srcEndPoint	Integer	0 – 0xFF	Source endpoint
cmd	Integer	0 – 0xFF	ZCL command id
specific	Bool	0 – 1	Frame Type 0 : Command is global for all clusters, including manufacturer specific clusters 1 : Command is specific or local to a cluster
disableDefaultRsp	Bool	0 – 1	Disable Default Response 0 : Default Response command will be return 1 : Default Response command will only be returned if there is an error
direction	Integer	0 – 1	Specifies the client/server direction for this command. 0 : sent from the client side 1 : sent from the server side
addrmode	Integer	1 – 2	1 : 16-bit group address for DstAddress, DstEndpoint not present 2 : 16-bit address for DstAddress and DstEndpoint present
manuCode	Integer	0 – 0xFFFF	Manufacturer code, not used manufacturer-specific flag is not set
cmdFormatLen	Integer	0 – 0xFFFF	Length of Frame Payload
cmdFormat	Various	-	Frame Payload

4.5.4 Simple description request

This primitive allow application layer to request simple description.

The semantics of this primitive are as flow:

- **Request**

```
typedef struct ZIGBEE_ZDO_SIMPLE_DESC_REQ_T
{
    uint16_t nwkAddr;
    uint8_t endpoint;
}zigbee_zdo_simple_desc_req_t;
```

Name	Type	Valid Range	Description
nwkAddr	Integer	0 – 0xFFFF	Netwok address
endpoint	Integer	0 – 0xFF	Acticve endpoint

4.5.5 Active endpoint request

This primitive allow application layer to request active endpoint.

The semantics of this primitive are as flow:

- **Request**

```
typedef struct ZIGBEE_ZDO_ACTIVE_EP_REQ_T
{
    uint16_t nwkAddr;
} zigbee_zdo_act_ep_req_t;
```

Name	Type	Valid Range	Description
nwkAddr	Integer	0 – 0xFFFF	Netwok address

4.5.6 Bind request

This primitive allow application layer to allow target to bind a group address
The semantics of this primitive are as flow:

- **Request**

```
typedef struct ZIGBEE_ZDO_BIND_REQ_T
{
    uint8_t bind;               // 0:unbind, 1:bind
    uint8_t srcEP;
    uint16_t clusterID;
    uint16_t reqDstAddr;
    uint16_t groupAddr;
    uint8_t srcIeeeAddr[8];
}zigbee_zdo_bind_req_t, zigbee_zdo_unbind_req_t;
```

Name	Type	Valid Range	Description
bind	Integer	0 –1	0 : unbind 1 : bind
srcEP	Integer	0 – 0xFF	The source endpoint for the binding entry.
clusterID	Integer	0 – 0xFFFF	The identifier of the cluster on the source device that is bound to the destination.
reqDstAddr	Integer	0 – 0xFF	The griup address of the request binding entry.
groupAddr	Integer	0 – 0xFFFF	The destination address for the
srcIeeeAddr	Various	-	The IEEE address for the source

4.5.7 Rejoin request

This primitive allows application layer to rejoin a network.

The semantics of this primitive are as flow:

- **Request**

  No parameter.

4.5.8 Node-Reset

This primitive announces application layer to reset all network configuration..

The semantics of this primitive are as flow:

- **Request**

  No parameter.

4.5.9 Permit join request

This primitive allow application layer to enable/disable permit join.

The semantics of this primitive are as flow:

- **Request**

```
typedef struct ZIGBEE_NWK_PERMIT_JOIN_REQ_T
{
    uint16_t enable;
    uint16_t timeout;
} zigbee_nwk_permit_join_req_t;
```

Name	Type	Valid Range	Description
enable	Integer	0 – 1	0 : Disable 1 : Enable
timeout	Integer	0 – 0xFFFF	Second of permit join timeout.

4.5.10 Network start indication

This primitive announce to application layer for network start staus.

The semantics of this primitive are as flow:

- **Indication**

```
typedef struct ZIGBEE_NWK_START_CFM_T
{
        uint32_t status;
}zigbee_nwk_start_cfm_t, zigbee_nwk_start_idc_t;
```

Name	Type	Valid Range	Description
status	Integer	0 – 0xFFFFFFFF	Coordinator: 0 : start a pan of network success Others : fail Router/End-deive: 0 : Scan and join a netwok success Others : fail

4.5.11 Device announce indication

This primitive announce to application layer for a node join in pan.

The semantics of this primitive are as flow:

- **Indication**

```
typedef struct ZIGBEE_ZDO_DEVICE_ANNOUNCE_IDC_T
{
        uint32_t shortAddr        :16;
        uint32_t capability       :8;
        uint8_t   ieeeAddr[8];
} zigbee_zdo_device_annce_idc_t;
```

Name	Type	Valid Range	Description
shortAddr	Integer	0 – 0xFFF0	Network address
capability	Integer	0 – 0xFF	Capability of the device
ieeeAddr	Various	-	The IEEE address

4.5.12 Active endpoint indication

This primitive announce to application layer for a actice endpoint response.

The semantics of this primitive are as flow:

- **Indication**

```
typedef struct ZIGBEE_ZDO_ACTIVE_EP_IDC_T
{
        uint8_t status;
        uint8_t epCounts;
        uint16_t nwkAddr;
        uint8_t ep_list[];
}zigbee_zdo_act_ep_idc_t;
```

Name	Type	Valid Range	Description
status	Integer	0 – 0xFF	Status of active endpoint request
epCounts	Integer	0 – 0xFF	The counts of active endpoint
nwkAddr	Integer	0 – 0xFFFF	The network address
ep_list	Various	-	The list of active endpoints

4.5.13 Simple description indication

This primitive announce to application layer for a simple description response.

The semantics of this primitive are as flow:

- **Indication**

```
typedef struct ZIGBEE_ZDO_SIMPLE_DESC_IDC_T
{
        uint8_t status;
        uint8_t endpoint;
        uint16_t nwkAddr;
        uint16_t profileID;
        uint16_t deviceID;
        uint16_t deviceVer;
        uint16_t in_cluster_count;
        uint16_t out_cluster_count;
        uint16_t clusterID[];
}zigbee_zdo_simple_desc_idc_t;
```

Name	Type	Valid Range	Description
status	Integer	0 – 0xFF	Status of simple description request
endpoint	Integer	0 – 0xFF	The active endpoint
nwkAddr	Integer	0 – 0xFFFF	Network address for the request
profileID	Integer	0 – 0xFFFF	Application profile identifier
deviceID	Integer	0 – 0xFFFF	Application device identifier
deviceVer	Integer	0 – 0xFFFF	Application device version
in_cluster_count	Integer	0 – 0xFF	Application input cluster count
out_cluster_count	Integer	0 – 0xFF	Application output cluster count
clusterID	Various	-	Input cluster identifier + Output cluster identifer

4.5.14 ZCL Data indication

This primitive announce to application layer for ZCL data.

The semantics of this primitive are as flow:

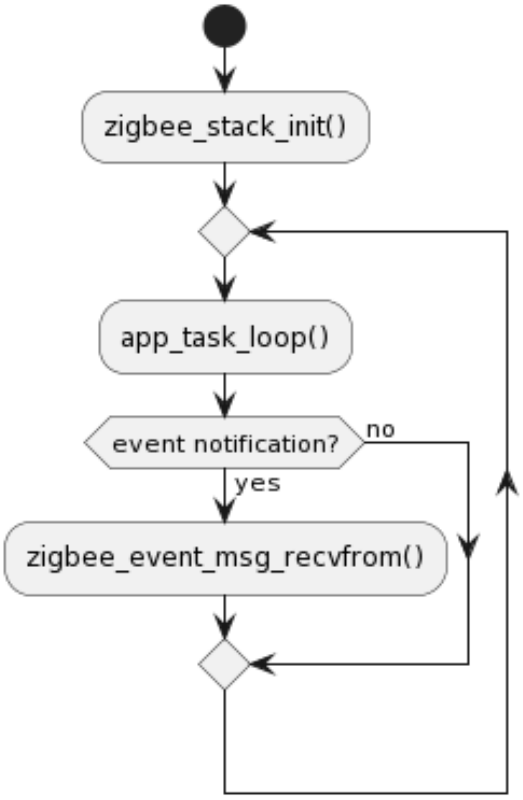
- **Indication**

```
typedef struct ZIGBEE_ZCL_DATA_REQ_T
{
    uint32_t dstAddr        : 16;
    uint32_t dstEndpint     : 8;
    uint32_t seq_num        : 8;

    uint32_t clusterID              : 16;
    uint32_t cmd                    : 8;
    uint32_t specific               : 1;
    uint32_t disableDefaultRsp      : 1;
    uint32_t direction              : 1;
    uint32_t                        : 5;

    uint32_t manuCode           : 16;
    uint32_t cmdFormatLen       : 16;

    uint8_t cmdFormat[];

}zigbee_zcl_data_req_t;
```

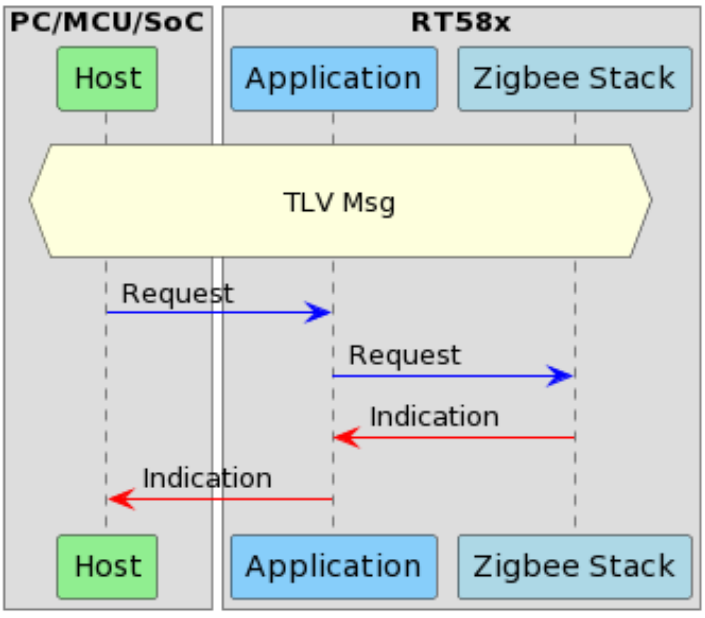
Name	Type	Valid Range	Description
dstAddr	Integer	0 – 0xFFFF	Destination address
dstEndpoint	Integer	0 – 0xFF	Destination endpoint
seq_num	Integer	0 – 0xFF	Sequence number
cmd	Integer	0 – 0xFF	ZCL command id
specific	Bool	0 – 1	Frame Type 0 : Command is global for all clusters, including manufacturer specific clusters 1 : Command is specific or local to a cluster
disableDefaultRsp	Bool	0 – 1	Disable Default Response 0 : Default Response command will be return 1 : Default Response command will only be returned if there is an error
direction	Integer	0 – 1	Specifies the client/server direction for this command. 0 : sent from the client side 1 : sent from the server side
manuCode	Integer	0 – 0xFFFF	Manufacturer code, not used manufacturer-specific flag is not set
cmdFormatLen	Integer	0 – 0xFFFF	Length of Frame Payload
cmdFormat	Various	-	Frame Payload

5. Zigbee Stack Library Setup Procedure

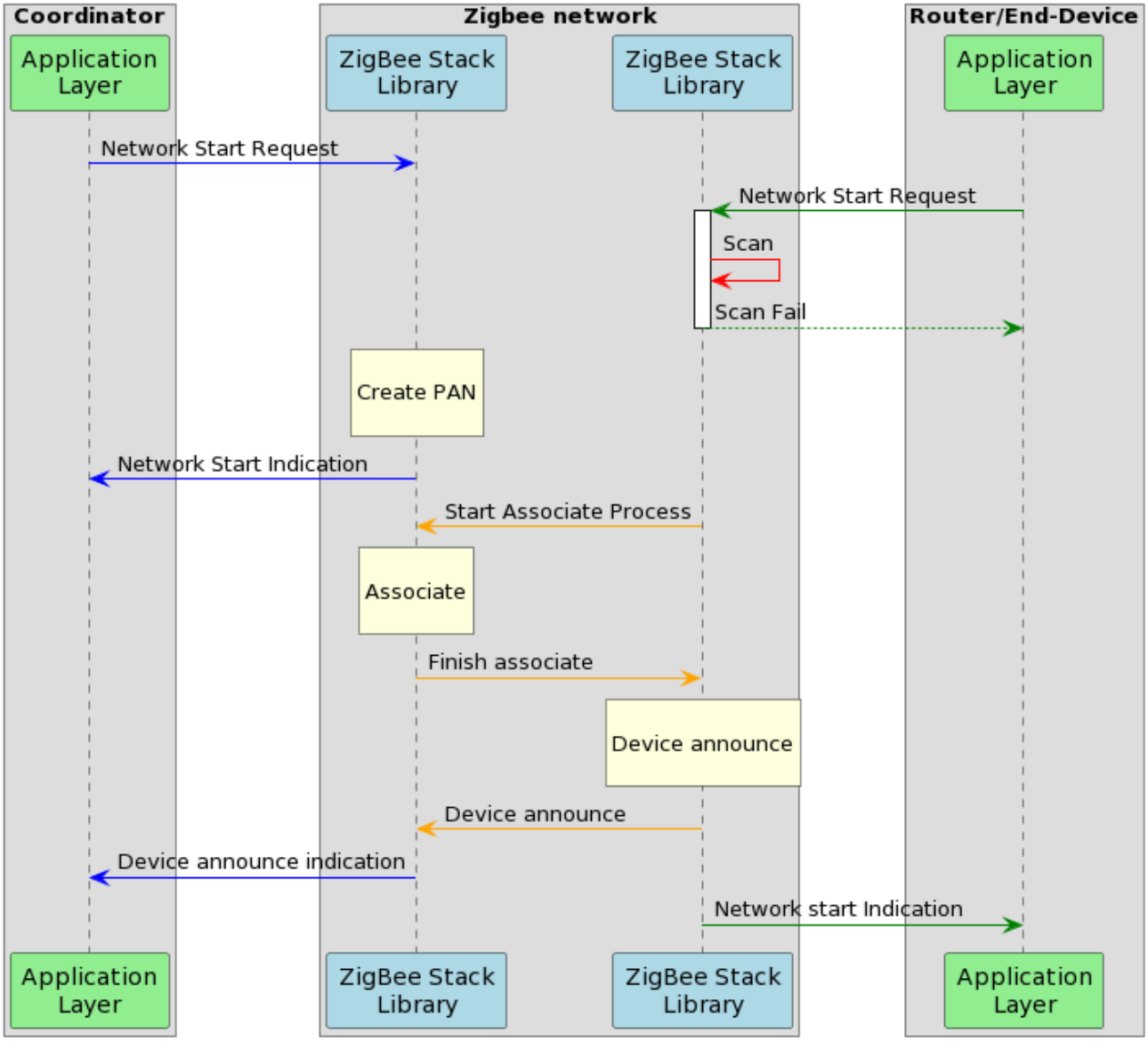
5.1 Initial and Message indication flow



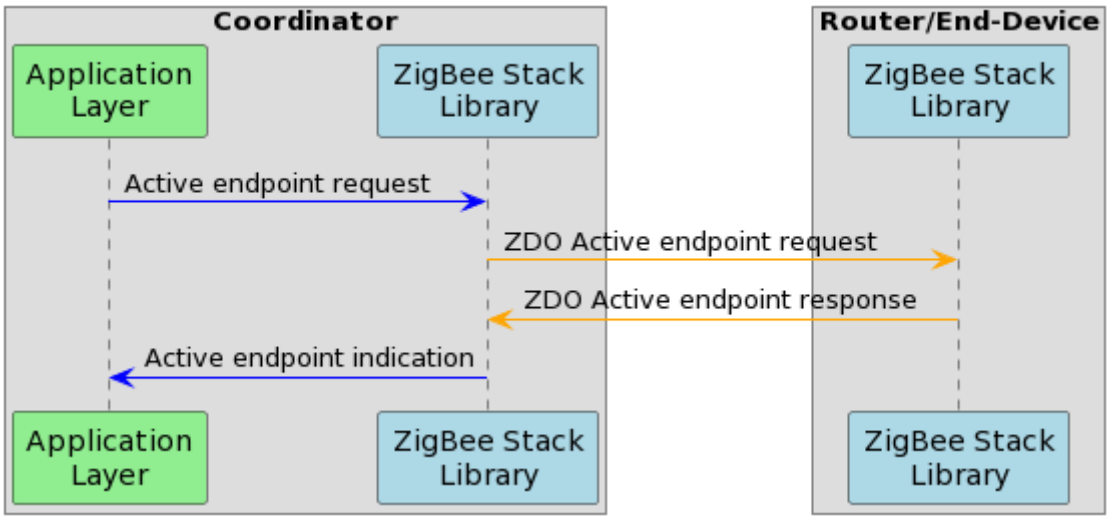
5.2 Message Sequence



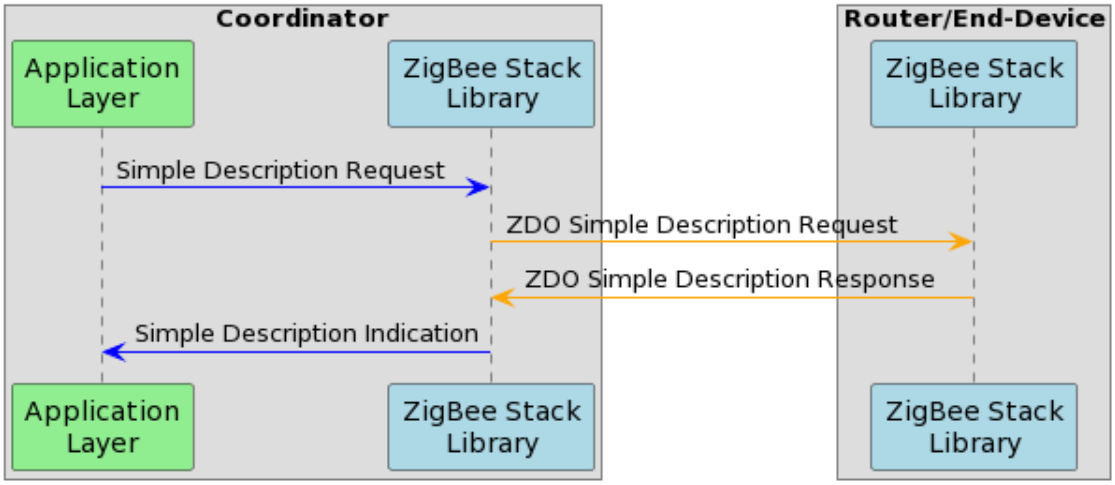
5.3 Network start Sequence



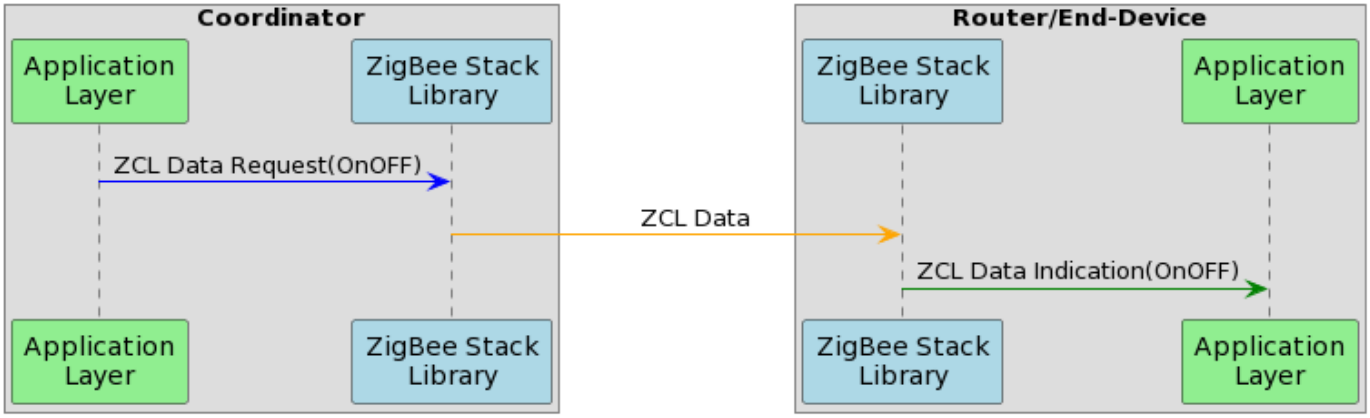
5.4 Active Endpoint Message Sequence



5.5 Simple Description Message Sequence



5.6 Generic OnOff Message Sequence



6. Zigbee ZCL Setup and Definitions

6.1 ZCL Attribute Data List Definitions

6.1.1 Basic

```
ZB_ZCL_DECLARE_BASIC_ATTRIB_LIST_EXT(attr_list, zcl_version,
  app_version, stack_version, hardware_version, manufacturer_name,
  model_id,date_code,  power_source, location_id, ph_env,
  sw_build_id)
```

Name	Description
attr_list	Attribute list name
zcl_version	Pointer to variable storing ZCL version
app_version	Pointer to the variable storing application version
stack_version	Pointer to the variable storing stack version
hardware_version	Pointer to the variable storing hardware version
manufacturer_name	Pointer to the variable storing manufacturer name
model_id	Pointer to the variable storing model identifier
date_code	Pointer to the variable storing date code
power_source	Pointer to variable storing power source attribute value
location_id	Pointer to variable storing location description attribute value
ph_env	Pointer to variable storing physical environment attribute value
sw_build_id	Pointer to the variable storing software version reference

6.1.2 Identify

```
ZB_ZCL_DECLARE_IDENTIFY_ATTRIB_LIST(attr_list, identify_time)
```

Name	Description
attr_list	Attribute list name
identify_time	Pointer to variable to store identify time

6.1.3 Group

```
ZB_ZCL_DECLARE_GROUPS_ATTRIB_LIST(attr_list, name_support)
```

Name	Description
attr_list	Attribute list name
name_support	Pointer to variable to store name_support attribute value

6.1.4 Scenes

```
ZB_ZCL_DECLARE_SCENES_ATTRIB_LIST(attr_list, scene_count,
  current_scene, current_group, scene_valid, name_support)
```

Name	Description
attr_list	Attribute list name
scene_count	Pointer to variable to store scene_count attribute value
current_scene	Pointer to variable to store current_scene attribute value
current_group	Pointer to variable to store current_group attribute value
scene_valid	Pointer to variable to store scene_valid attribute value
name_support	Pointer to variable to store name_suppor attribute value

6.1.5 OnOff

```
ZB_ZCL_DECLARE_ON_OFF_ATTRIB_LIST(attr_list, on_off)
```

Name	Description
attr_list	Attribute list name
on_off	Pointer to variable to store On/Off attribute value

6.1.6 Level Control

```
ZB_ZCL_DECLARE_LEVEL_CONTROL_ATTRIB_LIST(attr_list, current_level,
  remaining_time)
```

Name	Description
attr_list	Attribute list name
current_level	Pointer to variable to store current_level attribute value
remaining_time	Pointer to variable to store remaining_time attribute value

6.1.7 Color Control

```
ZB_ZCL_DECLARE_COLOR_CONTROL_ATTRIB_LIST(attr_list, current_X,
  current_Y)
```

Name	Description
attr_list	Attribute list name
current_X	Pointer to variable to store current_X attribute value
current_Y	Pointer to variable to store current_Y attribute value

6.2 ZCL Cluster Definitions

```
ZB_ZCL_CLUSTER_DESC(cluster_id, attr_count, attr_desc_list,
  cluster_role_mask, manuf_code)
```

Name	Description
cluster_id	ZCL 16-bit cluster id.
attr_count	Attributes number supported by the cluster
attr_desc_list	List of cluster attributes
cluster_role_mask	Cluster role
manuf_code	Manufacturer code for cluster and its attributes

6.3 Simple Description Definitions

```
ZB_DECLARE_SIMPLE_DESC(in_cluster_count, out_cluster_count)
```

Name	Description
in_cluster_count	Number of input clusters in descriptor
out_cluster_count	Number of output clusters in descriptor

```
ZB_AF_SIMPLE_DESC_TYPE (in_num, out_num) xxx_simple_desc = {
  endpoint, app_profile_id, app_device_id, app_device_version,
  app_input_cluster_count, app_output_cluster_count,
  app_cluster_list[in_num + out_num]
}
```

Name	Description
endpoint	Endpoint
app_profile_id	Application profile identifier
app_device_id	Application device identifier
app_device_version	Application device version
app_input_cluster_count	Application input cluster count
app_output_cluster_count	Application output cluster count
app_cluster_list	Application input and output cluster list

6.4 Endpoint Definitions

Initialize endpoint descriptor

```
ZB_AF_DECLARE_ENDPOINT_DESC(ep_name, ep_id, profile_id,
  reserved_length, reserved_ptr, cluster_number, cluster_list,
  simple_desc, rep_count, rep_ctx, level_ctrl_count, level_ctrl_ctx)
```

Name	Description
ep_name	endpoint name
ep_id	endpoint ID
profile_id	ID of profile deployed on this endpoint
reserved_length	unused parameter
reserved_ptr	unused parameter
cluster_number	number of clusters deployed on endpoint
cluster_list	pointer to cluster list
simple_desc	pointer to simple descriptor
rep_count	maximum number of attributes that are being reported on a device
rep_ctx	reporting context variable name (NULL if no reporting context)
level_ctrl_count	number of level control attributes
level_ctrl_ctx	level control context variable name (NULL if no level control context)

Initialize endpoint list

```
zb_af_endpoint_desc_t *xxx_ep_list[] = { &ep_name }
```

6.5 Device context

Contains a list of registered endpoints

Zigbee Stack Library initial will used this device context.

```
zb_af_device_ctx_t xxx_ctx = { ep_count, **ep_desc_list }
```

Name	Description
ep_count	Number of endpoints on device
**ep_desc_list	Endpoint list

7. Zigbee Stack Library Example Code

7.1 Ligiting device ZCL definition

```
//========================================================
//                Private Global Variables
//========================================================
/* Basic cluster attributes */
static const uint8_t attr_zcl_version  =
ZB_ZCL_BASIC_ZCL_VERSION_DEFAULT_VALUE;
static const uint8_t attr_app_version  = 0;
static const uint8_t attr_stack_version  = 0;
static const uint8_t attr_hw_version  = 0;
static const char attr_mf_name[]  = { 11, 'R', 'a', 'f', 'a', 'e', 'l', 'M',
'i', 'c', 'r', 'o'};
static const char attr_model_id[]  = { 9, 'L', 'i', 'g', 'h', 't', '0', '0',
'0', '1'};
static const char attr_date_code[]  = { 8, '2', '0', '2', '2', '0', '1', '2',
'1'};
static const char attr_sw_build_id[]  = { 8, '2', '0', '2', '2', '0', '1', '2',
'1'};
static const uint8_t attr_power_source =
ZB_ZCL_BASIC_POWER_SOURCE_DEFAULT_VALUE;

static uint8_t attr_location_id[] = { 16, ' ',' ',' ',' ',' ',' ',' ',' ',' '
',' ',' ',' ',' ',' ',' ',' ',' ' };
static uint8_t attr_ph_env = 0;
```

```
//=========================================================================
//                Global Variables
//=========================================================================
uint16_t g_attr_identify_time = 0;

/* Group cluster attributes data */
uint8_t g_attr_name_support = 0;

/* Scenes cluster attribute data */
uint8_t g_attr_scenes_scene_count;
uint8_t g_attr_scenes_current_scene;
uint8_t g_attr_scenes_scene_valid;
uint8_t g_attr_scenes_name_support;
uint16_t g_attr_scenes_current_group;

/* On/Off cluster attributes data */
uint32_t g_attr_on_off_on_off = ZB_ZCL_ON_OFF_ON_OFF_DEFAULT_VALUE;

/* Level cluster attributes data */
uint8_t g_attr_level_current_level = 0;
uint16_t g_attr_level_remaining_time = 0;

/* Color cluster attributes data */
uint16_t g_attr_color_current_x;
uint16_t g_attr_color_current_y;
//=========================================================================
//                Attribute definitions
//=========================================================================
ZB_ZCL_DECLARE_BASIC_ATTRIB_LIST_EXT(basic_attr_list,
    &attr_zcl_version,
    &attr_app_version,
    &attr_stack_version,
    &attr_hw_version,
    &attr_mf_name,
    &attr_model_id,
    &attr_date_code,
    &attr_power_source,
```

```
    &attr_location_id,
    &attr_ph_env,
    &attr_sw_build_id);

ZB_ZCL_DECLARE_IDENTIFY_ATTRIB_LIST(identify_attr_list,
    &g_attr_identify_time);

ZB_ZCL_DECLARE_GROUPS_ATTRIB_LIST(groups_attr_list,
    &g_attr_name_support);

ZB_ZCL_DECLARE_SCENES_ATTRIB_LIST(scenes_attr_list,
    &g_attr_scenes_scene_count,
    &g_attr_scenes_current_scene,
    &g_attr_scenes_current_group,
    &g_attr_scenes_scene_valid,
    &g_attr_scenes_name_support);

ZB_ZCL_DECLARE_ON_OFF_ATTRIB_LIST(on_off_attr_list,
    &g_attr_on_off_on_off);

ZB_ZCL_DECLARE_LEVEL_CONTROL_ATTRIB_LIST(level_control_attr_list,
    &g_attr_level_current_level,
    &g_attr_level_remaining_time);

ZB_ZCL_DECLARE_COLOR_CONTROL_ATTRIB_LIST(color_control_attr_list,
    &g_attr_color_current_x,
    &g_attr_color_current_y);

zb_zcl_cluster_desc_t g_zigbee_cluster_list[] =
{
    ZB_ZCL_CLUSTER_DESC(ZB_ZCL_CLUSTER_ID_BASIC,
        ZB_ZCL_ARRAY_SIZE(basic_attr_list, zb_zcl_attr_t),
        (basic_attr_list),
        ZB_ZCL_CLUSTER_SERVER_ROLE,
        ZB_ZCL_MANUF_CODE_INVALID
    ),
    ZB_ZCL_CLUSTER_DESC(ZB_ZCL_CLUSTER_ID_IDENTIFY,
```

```
        ZB_ZCL_ARRAY_SIZE(identify_attr_list, zb_zcl_attr_t),
        (identify_attr_list),
        ZB_ZCL_CLUSTER_SERVER_ROLE,
        ZB_ZCL_MANUF_CODE_INVALID
    ),
    ZB_ZCL_CLUSTER_DESC(ZB_ZCL_CLUSTER_ID_GROUPS,
        ZB_ZCL_ARRAY_SIZE(groups_attr_list, zb_zcl_attr_t),
        (groups_attr_list),
        ZB_ZCL_CLUSTER_SERVER_ROLE,
        ZB_ZCL_MANUF_CODE_INVALID
    ),
    ZB_ZCL_CLUSTER_DESC(ZB_ZCL_CLUSTER_ID_SCENES,
        ZB_ZCL_ARRAY_SIZE(scenes_attr_list, zb_zcl_attr_t),
        (scenes_attr_list),
        ZB_ZCL_CLUSTER_SERVER_ROLE,
        ZB_ZCL_MANUF_CODE_INVALID
    ),
    ZB_ZCL_CLUSTER_DESC(ZB_ZCL_CLUSTER_ID_ON_OFF,
        ZB_ZCL_ARRAY_SIZE(on_off_attr_list, zb_zcl_attr_t),
        (on_off_attr_list),
        ZB_ZCL_CLUSTER_SERVER_ROLE,
        ZB_ZCL_MANUF_CODE_INVALID
    ),
    ZB_ZCL_CLUSTER_DESC(ZB_ZCL_CLUSTER_ID_LEVEL_CONTROL,
        ZB_ZCL_ARRAY_SIZE(level_control_attr_list, zb_zcl_attr_t),
        (level_control_attr_list),
        ZB_ZCL_CLUSTER_SERVER_ROLE,
        ZB_ZCL_MANUF_CODE_INVALID
    ),
    ZB_ZCL_CLUSTER_DESC(ZB_ZCL_CLUSTER_ID_COLOR_CONTROL,
        ZB_ZCL_ARRAY_SIZE(color_control_attr_list, zb_zcl_attr_t),
        (color_control_attr_list),
        ZB_ZCL_CLUSTER_SERVER_ROLE,
        ZB_ZCL_MANUF_CODE_INVALID
    )
};
```

```
//=========================================================================
//                Simple desc definitions
//=========================================================================
ZB_DECLARE_SIMPLE_DESC(7, 0);

ZB_AF_SIMPLE_DESC_TYPE(7, 0) simple_desc_light =
{ 2,  ZB_AF_HA_PROFILE_ID,HA_COLOR_DIMMABLE_LIGHT_DEVICE_ID, 0, 0, 7, 0,
    {
        ZB_ZCL_CLUSTER_ID_BASIC,
        ZB_ZCL_CLUSTER_ID_IDENTIFY,
        ZB_ZCL_CLUSTER_ID_SCENES,
        ZB_ZCL_CLUSTER_ID_GROUPS,
        ZB_ZCL_CLUSTER_ID_ON_OFF,
        ZB_ZCL_CLUSTER_ID_LEVEL_CONTROL,
        ZB_ZCL_CLUSTER_ID_COLOR_CONTROL
    }
};
ZB_AF_DECLARE_ENDPOINT_DESC(light_ep, 2, ZB_AF_HA_PROFILE_ID, 0, NULL,
    ZB_ZCL_ARRAY_SIZE(g_zigbee_cluster_list, zb_zcl_cluster_desc_t),
    g_zigbee_cluster_list,
    (zb_af_simple_desc_1_1_t*)&simple_desc_light,
    0, NULL, 0, NULL);
zb_af_endpoint_desc_t *light_ep_list[] =
{
    &light_ep,
};
zb_af_device_ctx_t simple_desc_light_ctx =
{
    1,
    light_ep_list
};
```

7.2 Switch device ZCL definition

```
//=========================================================
//                Private Global Variables
//=========================================================
/* Basic cluster attributes */
static uint8_t attr_zcl_version  = ZB_ZCL_BASIC_ZCL_VERSION_DEFAULT_VALUE;
static uint8_t attr_power_source = ZB_ZCL_BASIC_POWER_SOURCE_DEFAULT_VALUE;

//=========================================================================
//                Attribute definitions
//=========================================================================
ZB_ZCL_DECLARE_BASIC_ATTRIB_LIST(
    basic_attr_list,
    &attr_zcl_version,
    &attr_power_source);

zb_zcl_cluster_desc_t g_zigbee_cluster_list[] =
{
    ZB_ZCL_CLUSTER_DESC(
        ZB_ZCL_CLUSTER_ID_BASIC,
        ZB_ZCL_ARRAY_SIZE(basic_attr_list, zb_zcl_attr_t),
        (basic_attr_list),
        ZB_ZCL_CLUSTER_SERVER_ROLE,
        ZB_ZCL_MANUF_CODE_INVALID
    ),
    ZB_ZCL_CLUSTER_DESC(
        ZB_ZCL_CLUSTER_ID_ON_OFF,
        0,
        NULL,
        ZB_ZCL_CLUSTER_CLIENT_ROLE,
        ZB_ZCL_MANUF_CODE_INVALID
    ),
    ZB_ZCL_CLUSTER_DESC(
        ZB_ZCL_CLUSTER_ID_LEVEL_CONTROL,
        0,
        NULL,
```

```
        ZB_ZCL_CLUSTER_CLIENT_ROLE,
        ZB_ZCL_MANUF_CODE_INVALID
    ),
    ZB_ZCL_CLUSTER_DESC(
        ZB_ZCL_CLUSTER_ID_COLOR_CONTROL,
        0,
        NULL,
        ZB_ZCL_CLUSTER_CLIENT_ROLE,
        ZB_ZCL_MANUF_CODE_INVALID
    ),

};
//==========================================================================
//                Simple desc definitions
//==========================================================================
ZB_DECLARE_SIMPLE_DESC(0, 1);

ZB_AF_SIMPLE_DESC_TYPE(1, 1) simple_desc_switch_ep_1 =
{
    BUTTON_1_EP, ZB_AF_HA_PROFILE_ID, HA_COLOR_DIMMER_SWITCH_DEVICE_ID,
    0, 0, 1, 1,
    {
        ZB_ZCL_CLUSTER_ID_BASIC,
        ZB_ZCL_CLUSTER_ID_ON_OFF,
    }
};

ZB_AF_SIMPLE_DESC_TYPE(0, 1) simple_desc_switch_ep_2 =
{
    BUTTON_2_EP, ZB_AF_HA_PROFILE_ID, HA_DIMMER_SWITCH_DEVICE_ID,
    0, 0, 0, 1,
    {
        ZB_ZCL_CLUSTER_ID_LEVEL_CONTROL,
    }
};
```

```
ZB_AF_SIMPLE_DESC_TYPE(0, 1) simple_desc_switch_ep_3 =
{
    BUTTON_3_EP, ZB_AF_HA_PROFILE_ID, HA_COLOR_DIMMER_SWITCH_DEVICE_ID,
    0, 0, 0, 1,
    {
        ZB_ZCL_CLUSTER_ID_COLOR_CONTROL
    }
};
ZB_AF_DECLARE_ENDPOINT_DESC(
    switch_ep_1, BUTTON_1_EP, ZB_AF_HA_PROFILE_ID, 0, NULL,
    ZB_ZCL_ARRAY_SIZE(g_zigbee_cluster_list, zb_zcl_cluster_desc_t),
    g_zigbee_cluster_list,
    (zb_af_simple_desc_1_1_t*)&simple_desc_switch_ep_1,
    0, NULL, 0, NULL);
ZB_AF_DECLARE_ENDPOINT_DESC(
    switch_ep_2, BUTTON_2_EP, ZB_AF_HA_PROFILE_ID, 0, NULL,
    ZB_ZCL_ARRAY_SIZE(g_zigbee_cluster_list, zb_zcl_cluster_desc_t),
    g_zigbee_cluster_list,
    (zb_af_simple_desc_1_1_t*)&simple_desc_switch_ep_2,
    0, NULL, 0, NULL);
ZB_AF_DECLARE_ENDPOINT_DESC(
    switch_ep_3, BUTTON_3_EP, ZB_AF_HA_PROFILE_ID, 0, NULL,
    ZB_ZCL_ARRAY_SIZE(g_zigbee_cluster_list, zb_zcl_cluster_desc_t),
    g_zigbee_cluster_list,
    (zb_af_simple_desc_1_1_t*)&simple_desc_switch_ep_3,
    0, NULL, 0, NULL);
zb_af_endpoint_desc_t *ep_list_switch[] =
{
    &switch_ep_1,
    &switch_ep_2,
    &switch_ep_3,
};
zb_af_device_ctx_t simple_desc_switch_ctx =
{
    3,
    ep_list_switch
};
```

7.3 Zigbee stack Initial

```
void zigbee_app_init(void)
{
    /* Initil LED, Button, Console or UART */
    bsp_init(BSP_INIT_DEBUG_CONSOLE, NULL);
    bsp_init((BSP_INIT_LEDS | BSP_INIT_BUTTONS), app_bsp_event_handle);

    /* Retarget stdout for utility & initial utility logging */
    utility_register_stdout(bsp_console_stdout_char,
                            bsp_console_stdout_string);
    util_log_init();

    util_log_on(UTIL_LOG_PROTOCOL);

    gt_app_cfg.p_zigbee_device_contex_t = &simple_desc_switch_ctx;
    gt_app_cfg.pf_evt_indication = app_evt_indication_cb;

    info_color(LOG_BLUE, "Initial ZigBee stack\n");
    zigbee_stack_init(&gt_app_cfg);

    sys_queue_new(&app_msg_q, 16, sizeof(app_queue_t));

    info("Create app task\n");
    sys_task_new("app", app_main_task, NULL, 128, TASK_PRIORITY_APP);
}
```

7.4 Event Message receive

```
static void app_evt_indication_cb(uint32_t data_len)
{
    int i32_err;
    uint8_t *pBuf = sys_malloc(data_len);
    app_queue_t t_app_q;
    do
    {
        if(!pBuf)
            break;
        t_app_q.event = 0;
        i32_err = zigbee_event_msg_recvfrom(pBuf, &data_len);
        t_app_q.pt_tlv = (sys_tlv_t *)pBuf;
        if (i32_err == 0)
        {
            sys_queue_send_with_timeout(&app_msg_q, &t_app_q, 0);
        }
        else
        {
            info_color(LOG_RED, "[%s] sys_err = %d !\n", __func__, i32_err);
            sys_free(pBuf);
        }
    } while (0);
}
```

7.5 Event Message send

```
uint32_t zigbee_nwk_start_request(uint32_t device_role, uint32_t
channel_mask, uint32_t maxChild, uint16_t panID)
{
    sys_tlv_t *pt_tlv;
    zigbee_nwk_start_req_t *pt_nwk_start_req;
    uint8_t ieeeAddr[8] = {0x0a, 0x0b, 0x0c, 0x0d, 0x0e, 0x0f, 0x10, 0x10};
    uint32_t cfm_status = 0xFFFF;
    uint32_t rand;

    do
    {
        if(device_role == ZIGBEE_DEVICE_ROLE_CORDINATOR)
            ieeeAddr[7] = 0x10;
        else if(device_role == ZIGBEE_DEVICE_ROLE_ROUTER)
            ieeeAddr[7] = 0x11;
        else
        {
            info_color(LOG_RED, "Not support role\n");
            break;
        }

        pt_tlv = sys_malloc(SYS_TLV_HEADER_SIZE +
                            sizeof(zigbee_nwk_start_req_t));
        if(!pt_tlv)
            break;
        pt_tlv->type = ZIGBEE_EVT_TYPE_NWK_START_REQ;
        pt_tlv->length = sizeof(zigbee_nwk_start_req_t);
        pt_nwk_start_req = (zigbee_nwk_start_req_t *)pt_tlv->value;
        pt_nwk_start_req->deviceRole = device_role;
        pt_nwk_start_req->channelMask = channel_mask;
        pt_nwk_start_req->maxChild = maxChild;
        pt_nwk_start_req->PANID = panID;
        memcpy(pt_nwk_start_req->ieeeAddr, ieeeAddr, 8);
        cfm_status = zigbee_event_msg_sendto(pt_tlv);
```

```
        sys_free(pt_tlv);
    } while (0);

    return cfm_status;
}
```

Revision History

Revision	Description	Owner	Date
1.0	Initial version	Rex	2022/01/21