Mindtree
Welcome to possible

EtherMind Mesh

Application Developer's Guide

Version 0.4 | 31 Aug 2018

COPYRIGHT INFORMATION

REVISION HISTORY

Ver	Change Description	Date	Author	Reviewer
0.1	Initial Draft	10-April-2018	EtherMind Team	EtherMind Team
0.2	Updated multiple sections	20-May-2018	EtherMind Team	EtherMind Team
0.3	Updated document format	30-May-2018	EtherMind Team	EtherMind Team
0.4	Updated Friendship code segment for LPN	31-Aug-2018	EtherMind Team	EtherMind Teams

Table of Contents

Introduction

The EtherMind Mesh Core Stack & Models provide highly flexible and feature rich APIs for application developer to use and develop portable application for a variety of Bluetooth enabled devices and accessories.

This is the generic version of the EtherMind Mesh Developer's Guide document. Content in this document is written generically to address application development in any platform, or, operating systems on which EtherMind Mesh Core Stack & Models are supported.

Document Purpose

The purpose of this document is to provide an application developer, with existing knowledge of the Bluetooth Mesh wireless technology, with guidelines for developing application using EtherMind Mesh Core Stack & Models APIs. This document serves as a central point of information and as a starting guide for the application developers, and, complements the EtherMind Mesh API (Core Mesh Stack & Models) Document ([1]).

What This Document Covers

This document is organized in a manner most suitable for the application developer to read and understand. The key concepts are described in step-by-step manner, separated by chapters, as described below:

- ❑Important points to consider prior to writing application using EtherMind Mesh APIs.
- ❑How to initialize the Mesh Core Stack and Model(s)
- ❑How to use Mesh Core and Model (Client and Server) APIs & Callbacks
- ❑ How to create own Model

Document Symbols and Conventions

In this document the following symbols and conventions are used.

An important "note" is stated as follows:

✍ **Note**

✍ This is an important note concerning …

Additional references & reading are documented as follows:

See Also

- An example of application calling `MS_xyz()` API.
- Refer to XYZ Document for further details …

Sample code segment and examples are documented as follows:

Example: Sample Code for …

```
/*
 *  This is an example/sample code segment
 */
```

Following arrow conventions are used in signalling/message sequence charts:

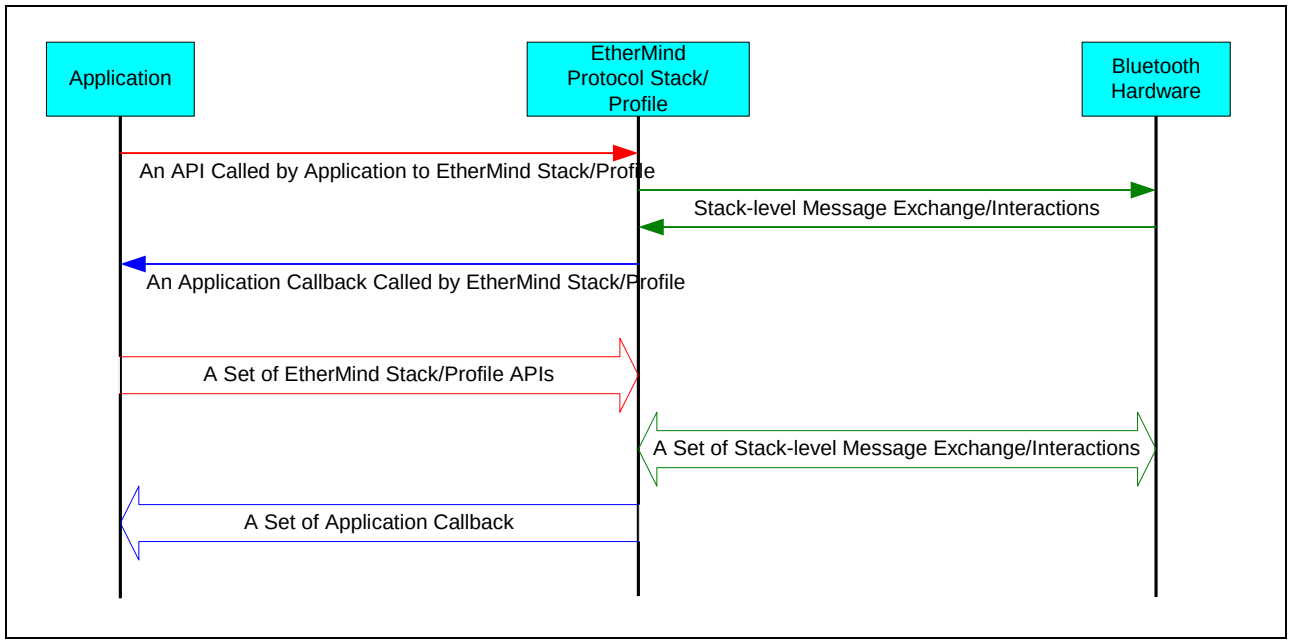


Figure 1: Arrow Convention

Section 1: Setting Up the Development Environment

Depending on the target platform, an application developer can make use of any available IDE of choice to build application using EtherMind Mesh Stack & Model APIs. However there are certain important considerations that need to be understood prior to starting to write an application.

Configuring EtherMind Mesh Stack & Module Modules

The EtherMind Mesh Core Protocol & Model stack is a highly portable and configurable stack. Two most important of the many configuration options provided by the EtherMind Mesh Stack are as stated below:

- Compilation Flags
- Tunable Constants

EtherMind Mesh Compilation Flags

The purposes of the EtherMind Mesh Compilation Flags, or Switches, are as follows:

- To enable/disable inclusion of a stack module
- To enable/disable/control the inclusion of features provided by various stack modules
- To choose the platform, operating system etc. for which the stack to be compiled
- To choose the compliance to specific architectural design, or, Bluetooth Mesh specification version

These compilation flags are specified in the EtherMind Mesh Header File `MS_features.h`.

Example: EtherMind Mesh Compilation Flags

```
/*
 *  MS_RELAY_SUPPORT
 *
 *  This flag is used to enable support for Relay feature.
 *
 *  Dependency: None.
 */
#define MS_RELAY_SUPPORT
```

Note

- Inclusion or non-inclusion of EtherMind Compilation Flags controls EtherMind Mesh Stack & Model module's code size (ROM) requirement. Hence, it is recommended that correct set of required compilation flags should be decided to limit features included in the final executable.
- Decision on correct set of EtherMind Mesh Compilation Flags is required only when the application developer has access to the EtherMind Mesh Stack & Model Source Codes, and/or, the same needs to be built/compiled before application could be linked.
- When the application developer does not have access to the EtherMind Stack & Profile Source Codes, and/or, the same has been supplied in pre-built object code format (DLL or Library or Object module), the application should be built and linked to supplied EtherMind object code using the same set of EtherMind Mesh Compilation Flags that have been used to build the EtherMind object codes.

EtherMind Mesh Tunable Constants

Various EtherMind Mesh Stack modules use Tunable Constant definitions to control limits of Queues, Buffer Size etc., which can be configured during the time of compilation and build of the Stack & Model modules. These tunable constants are specified in the EtherMind Mesh Header File `MS_limits.h`.

See Also

- Refer to `MS_limits.h` for details on the available tunable constants for a module and understand their scope and purpose. Certain tunable constants are inter-dependent, and the care should be taken while configuring values for the same.

In general, each tunable constant is preceded with a code comment block containing the following information:

- Name, purpose, description & background of the tunable constant
- Description of dependency of this tunable constant on other tunable constant
- Recommended minimum & maximum values for the tunable constant

An example of an EtherMind Mesh Tunable Constant is provided for reference below:

Example: EtherMind Mesh Tunable Constants

```
/*
 *  Maximum number of subnets the device can store information about.
 *
 *  Minimum Value: 1
 *  Maximum Value: can be anything.
 */
#define MS_MAX_SUBNETS                                  10
```

✍ **Note**

- ✍ Configuration of Tunable Constants controls EtherMind Mesh Stack & Model module's data size (RAM) requirement.
- ✍ 'MS_HAVE_DYNAMIC_CONFIG' feature flag controls if the application developer can change the Tunable Constants and have the desired configuration on a pre-built library of EtherMind Mesh Stack & Model.
- ✍ If 'MS_HAVE_DYNAMIC_CONFIG' feature flag is defined
  - Application Developer can change the Tunable Constants which are part of the MS_CONFIG data structure, defined in MS_common.h, even if the EtherMind Mesh Stack & Model is supplied in pre-built object code format (DLL or Library module)
- ✍ If 'MS_HAVE_DYNAMIC_CONFIG' feature flag is not defined
  - Configuration of Tunable Constants is required only when the application developer has access to the EtherMind Mesh Stack & Model Source Codes, and/or, the same needs to be built/compiled before application could be linked.
  - Reconfiguration of Tunable Constants shall not be done when the application developer does not have access to the EtherMind Mesh Stack & Model Source Codes, and/or, the same has been supplied in pre-built object code format (DLL or Library module).

The available compilation flags/switches are specified and described in detail, including dependencies, in the MS_features.h and MS_limits.h header files.

Inclusion of EtherMind Header Files

The table below describes various EtherMind Mesh Header Files, which may be included in application source files, with their purposes.

EtherMind Header File	Comments/Purpose/Description
MS_common.h	This is the most important EtherMind Mesh header file, and, must be included by every application source file before including any other EtherMind Mesh/Model header files. This header file, in turn, includes several other important and required EtherMind Mesh header files, some of which are described below: • EM_os.h - EtherMind OS Abstraction • MS_common_pl.h - EtherMind Mesh Platform Abstraction • MS_limits.h - EtherMind Mesh Tunable Constants • MS_assigned_numbers.h - Bluetooth SIG defined Assigned Numbers • MS_error.h - EtherMind Mesh API/Callback Result/Error Code Definitions
Mesh Core API Header Files: MS_brr_api.h MS_prov_api.h MS_net_api.h MS_access_api.h etc.	Depending on application's need some or all of these API header files may be needed to be included in application source code files.

EtherMind Header File	Comments/Purpose/Description

Mesh Model API Header Files:
`MS_config_api.h`
`MS_health_server_api.h`
`MS_health_client_api.h`
`MS_generic_onoff_api.h`
etc.
	Depending on application's need some or all of these API header files may be needed to be included in application source code files.

EtherMind Header Files Include Directories

The development tool, which is being used to compile/build the application, should be configured to add the following directories/paths for the additional (header file) include directories. This facilitates the development tool to search for the EtherMind header files during compilation of the application source codes.

- EtherMind Mesh Common Header Files Directory:

  `mesh/export/include`

- EtherMind Platform/Operating System specific Header Files Directory:
  - `osal/src/<your-os>`
  - `platforms`

All paths mentioned above are relative to the directory where EtherMind Mesh package are installed by the application developer.

✍ **Note**

✍ All paths mentioned above are relative to the directory where EtherMind source codes are installed by the application developer.

Section 2: Initializing EtherMind Mesh Stack

EtherMind Mesh Stack exposes interfaces to initialize Core modules and Models. Depending on the platform/environment, before the Mesh initialization is done, it might be required to initialize the underlying operating system, debug/timer module, non-volatile storage etc.

The `MS_init()` is the first Mesh API that the application must call to initialize the entire EtherMind Mesh Core Stack. During this initialization, various modules create and initialize their respective synchronization and conditional variables and allocate any dynamic memory (if required). All modules perform platform level initialization during this process.

Internally, `MS_init()` calls the initialization routine of each module, one after the other. It follows a bottom-up approach, i.e., the lower layers are initialized before the higher layers.

Mesh stack does not create any thread/task. EtherMind mesh stack is a non-blocking implementation, with all the interactions with the mesh stack will be through registered function callback. Application can invoke the `MS_init()` from any of the thread/task it has access to, like from application thread/task itself.

If '`MS_HAVE_DYNAMIC_CONFIG`' feature flag is defined, the application must initialize a MS_CONFIG variable and pass the same to `MS_init()`. Application can use utility macro `MS_INIT_CONFIG()` to initialize the Tunable parameters with the values defined in `MS_limits.h` file.

Application is required to include the "`MS_common.h`" to be able to invoke `MS_init()`. An example of application calling `MS_init()` is shown below:

Listing 1: EtherMind Mesh Initialization Example

```
#include "MS_common.h"

int main (int argc, char **argv)
{
    MS_CONFIG * config_ptr;

#ifdef MS_HAVE_DYNAMIC_CONFIG
    MS_CONFIG  config;

    /* Initialize dynamic configuration */
    MS_INIT_CONFIG(config);

    config_ptr = &config;
#else
    config_ptr = NULL;
#endif /* MS_HAVE_DYNAMIC_CONFIG */

    /* Initialize OSAL */
    EM_os_init();

    /* Initialize Debug Module */
    EM_debug_init();
```

Listing 1: EtherMind Mesh Initialization Example

```
    /* Initialize Timer Module */
    EM_timer_init();
    timer_em_init();

    /* Initialize utilities */
    nvsto_init();

    /* Initialize Mesh Stack */
    MS_init(config_ptr);

    /* Register with underlying BLE stack */
    blebrr_register();


    ...


    return 0;
}
```

See Also

- An example of application calling `MS_init()` API.
- A detailed description of the Mesh APIs can be found in EtherMind Mesh Stack API [2].

Note

- The EtherMind Stack modules may behave in unpredictable manner if `MS_init()` is not invoked to initialize the EtherMind Mesh Core Stack prior to calling any other EtherMind Mesh Stack API(s).
- EtherMind Mesh Models including Foundation Models (Configuration and Health Models) are not initialized during the `MS_init()` API. Initialization of required EtherMind Mesh Models needs to be done by application once `MS_init()` returns successfully.
- There is no interface exposed to shutdown Mesh stack. If there are specific usecase which requires shutting down Mesh stack, when the system is still powered on, such interface can be introduced in future.

Section 3: Demonstration Applications

How to write a Model server application?

The APIs exported by various EtherMind Mesh Models are available and described in detail in their respective "Model" API Documentation Sections [such as, the "EtherMind Mesh Configuration Model API s", "EtherMind Mesh Generic OnOff APIs" etc.].

Most of the EtherMind Mesh Models follow common design principle for exporting APIs for application to use. In particular, all EtherMind Mesh Models provide APIs to make Initialize, set publication and subscription and exchange messages (set/get/status etc.). In addition, all Mesh Models provide methods for registering their respective Callback function mechanism to inform application of result/response of various APIs, and asynchronous events, such as reception of request and/or response messages.

Generic OnOff Model is used below as an example to explore the usage of Mesh APIs.

Generic OnOff Server Application

Generic OnOff Sever application uses following set of APIs

[1] Initialization of Core Mesh Stack

[2] Initialization of Foundation Model

[3] Initializations of Generic OnOff Server

[4] Provision APIs for provisionee (if the device is not already provisioned)

Generic OnOff Server example for Advertising Bearer.

✍ **Note**

✍ Steps [1], [2] and [4] will be the same for all model server implementation.

Listing 2: Header File Inclusion for Foundation and Generic OnOff Models

```
/* ---------------------------------------- Header File Inclusion */
#include "MS_common.h"
#include "MS_access_api.h"
#include "MS_config_api.h"
#include "MS_health_server_api.h"
#include "MS_generic_onoff_api.h"

/* Console Input/Output */
#define CONSOLE_OUT(...)    printf(__VA_ARGS__)
#define CONSOLE_IN(...)     scanf(__VA_ARGS__)
```

Listing 3: Foundation Models Initialization and helper routines Example

```
/* Model Server - Foundation Models */

/* Health Server - Test Routines */
static void UI_health_self_test_00(UINT8 test_id, UINT16 company_id)
{
}

static void UI_health_self_test_01(UINT8 test_id, UINT16 company_id)
{
}

static void UI_health_self_test_FF(UINT8 test_id, UINT16 company_id)
{
}

/* List of Self Tests */
static MS_HEALTH_SERVER_SELF_TEST UI_health_server_self_tests[] =
{
        {
            0x00, /* Test ID: 0x00 */
            UI_health_self_test_00
        },
        {
            0x01, /* Test ID: 0x01 */
            UI_health_self_test_01
        },
        {
            0xFF, /* Test ID: 0xFF */
            UI_health_self_test_FF
        }
};

/**
 * \brief Health Server application Asynchronous Notification Callback.
 *
 * \par Description
 * Health Server calls the registered callback to indicate events occurred to the
 * application.
 *
 * \param handle        Model Handle.
 * \param opcode        Opcode.
 * \param data_param    Data associated with the event if any or NULL.
 * \param data_len      Size of the event data. 0 if event data is NULL.
 */
API_RESULT UI_health_server_cb
           (
               MS_ACCESS_MODEL_HANDLE * handle,
               UINT32                   opcode,
               UCHAR                  * data_param,
               UINT16                   data_len
           )
{
    CONSOLE_OUT(
    "Health Server Callback. Not handled. Returning\n");

    return API_SUCCESS;
}
```

Listing 3: Foundation Models Initialization and helper routines Example

```
API_RESULT UI_register_foundation_model_servers
           (
               MS_ACCESS_ELEMENT_HANDLE element_handle
           )
{
    /* Configuration Server */
    MS_ACCESS_MODEL_HANDLE   UI_config_server_model_handle;
    MS_ACCESS_MODEL_HANDLE   UI_health_server_model_handle;
    API_RESULT retval;

    /* Health Server */
    UINT16                       company_id;
    MS_HEALTH_SERVER_SELF_TEST * self_tests;
    UINT32                       num_self_tests;

    CONSOLE_OUT("In Model Server - Foundation Models\n");

    retval = MS_config_server_init(element_handle, &UI_config_server_model_handle);

    CONSOLE_OUT("Config Model Server Registration Status: 0x%04X\n", retval);

    /* Health Server */
    company_id = 0x0000;
    self_tests = &UI_health_server_self_tests[0];
    num_self_tests =
sizeof(UI_health_server_self_tests)/sizeof(MS_HEALTH_SERVER_SELF_TEST);

    retval = MS_health_server_init
             (
                 element_handle,
                 &UI_health_server_model_handle,
                 company_id,
                 self_tests,
                 num_self_tests,
                 UI_health_server_cb
             );

    if (API_SUCCESS == retval)
    {
        CONSOLE_OUT(
        "Health Server Initialized. Model Handle: 0x%04X\n",
        UI_health_server_model_handle);
    }
    else
    {
        CONSOLE_OUT(
        "[ERR] Sensor Server Initialization Failed. Result: 0x%04X\n",
        retval);
    }

    return retval;
}
```

Listing 4: Generic OnOff Model state initialization and Get/Set state handlers Example

```
/* ---- Generic OnOff States and Get/Set Handlers */
static MS_STATE_GENERIC_ONOFF_STRUCT UI_generic_onoff;

/* Generic OnOff Model state Initialization */
void UI_generic_onoff_model_states_initialization(void)
{
    EM_mem_set(&UI_generic_onoff, 0, sizeof(UI_generic_onoff));
}

/* Generic OnOff Model Get Handler */
API_RESULT UI_generic_onoff_model_state_get(UINT16 state_t, UINT16 state_inst, void *
param, UINT8 direction)
{
    API_RESULT retval;

    retval = API_SUCCESS;

    switch(state_t)
    {
        case MS_STATE_GENERIC_ONOFF_T:
        {
            MS_STATE_GENERIC_ONOFF_STRUCT * param_p;

            param_p = (MS_STATE_GENERIC_ONOFF_STRUCT *)param;

            /* Ignoring Instance and direction right now */
            *param_p = UI_generic_onoff;
        }
        break;

        default:
        break;
    }

    return retval;
}

/* Generic OnOff Model Set Handler */
API_RESULT UI_generic_onoff_model_state_set(UINT16 state_t, UINT16 state_inst, void *
param, UINT8 direction)
{
    API_RESULT retval;

    retval = API_SUCCESS;
    switch (state_t)
    {
        case MS_STATE_GENERIC_ONOFF_T:
        {
            MS_STATE_GENERIC_ONOFF_STRUCT * param_p;

            param_p = (MS_STATE_GENERIC_ONOFF_STRUCT *)param;

            /* Instantaneous Change */
            UI_generic_onoff.onoff = param_p->onoff;

            *param_p = UI_generic_onoff;
            CONSOLE_OUT("[state] current: 0x%02X\n", UI_generic_onoff.onoff);
```

Listing 4: Generic OnOff Model state initialization and Get/Set state handlers Example

```
            CONSOLE_OUT("[state] target: 0x%02X\n", UI_generic_onoff.target_onoff);
            CONSOLE_OUT("[state] remaining_time: 0x%02X\n",
UI_generic_onoff.transition_time);

            /* Ignoring Instance and direction right now */
        }
        break;

        default:
        break;
    }

    return retval;
}

/* Model state Initialization */
void UI_model_states_initialization(void)
{
    /* Generic OnOff States */
    UI_generic_onoff_model_states_initialization();
}
```

Listing 5: Generic OnOff Model Initialization and Callback routine Example

```
/* Generic OnOff Model Server */
/**
 * \brief Server Application Asynchronous Notification Callback.
 *
 * \par Description
 * Generic_Onoff server calls the registered callback to indicate events occurred to the
application.
 *
 * \param [in] ctx           Context of message received for a specific model instance.
 * \param [in] msg_raw       Uninterpreted/raw received message.
 * \param [in] req_type      Requested message type.
 * \param [in] state_params  Model specific state parameters.
 * \param [in] ext_params    Additional parameters.
 */
API_RESULT UI_generic_onoff_server_cb
           (
               /* IN */ MS_ACCESS_MODEL_REQ_MSG_CONTEXT    * ctx,
               /* IN */ MS_ACCESS_MODEL_REQ_MSG_RAW        * msg_raw,
               /* IN */ MS_ACCESS_MODEL_REQ_MSG_T          * req_type,
               /* IN */ MS_ACCESS_MODEL_STATE_PARAMS       * state_params,
               /* IN */ MS_ACCESS_MODEL_EXT_PARAMS         * ext_params
           )
{
    MS_STATE_GENERIC_ONOFF_STRUCT    param;
    MS_ACCESS_MODEL_STATE_PARAMS     current_state_params;
    API_RESULT                       retval;

    retval = API_SUCCESS;

    /* Check message type */
    if (MS_ACCESS_MODEL_REQ_MSG_T_GET == req_type->type)
```

Listing 5: Generic OnOff Model Initialization and Callback routine Example

```
    {
        CONSOLE_OUT("[GENERIC_ONOFF] GET Request.\n");

        UI_generic_onoff_model_state_get(state_params->state_type, 0, &param, 0);

        current_state_params.state_type = state_params->state_type;
        current_state_params.state = &param;

        /* Using same as target state and remaining time as 0 */
    }
    else if (MS_ACCESS_MODEL_REQ_MSG_T_SET == req_type->type)
    {
        CONSOLE_OUT("[GENERIC_ONOFF] SET Request.\n");

        retval = UI_generic_onoff_model_state_set(state_params->state_type, 0,
(MS_STATE_GENERIC_ONOFF_STRUCT *)state_params->state, 0);

        current_state_params.state_type = state_params->state_type;
        current_state_params.state = (MS_STATE_GENERIC_ONOFF_STRUCT *)state_params->state;
    }

    /* See if to be acknowledged */
    if (0x01 == req_type->to_be_acked)
    {
        CONSOLE_OUT("[GENERIC_ONOFF] Sending Response.\n");

        /* Parameters: Request Context, Current State, Target State (NULL: to be ignored),
Remaining Time (0: to be ignored), Additional Parameters (NULL: to be ignored) */
        retval = MS_generic_onoff_server_state_update(ctx, &current_state_params, NULL, 0,
NULL);
    }

    return retval;
}

API_RESULT UI_register_generic_onoff_model_server
           (
               MS_ACCESS_ELEMENT_HANDLE element_handle
           )
{
    /* Generic OnOff Server */
    MS_ACCESS_MODEL_HANDLE   UI_generic_onoff_server_model_handle;
    API_RESULT retval;

    CONSOLE_OUT("In Generic OnOff Model Server\n");

    retval = MS_generic_onoff_server_init
             (
                 element_handle,
                 &UI_generic_onoff_server_model_handle,
                 UI_generic_onoff_server_cb
             );

    if (API_SUCCESS == retval)
    {
        CONSOLE_OUT(
        "Generic Onoff Server Initialized. Model Handle: 0x%04X\n",
```

Listing 5: Generic OnOff Model Initialization and Callback routine Example

```
        UI_generic_onoff_server_model_handle);
    }
    else
    {
        CONSOLE_OUT(
        "[ERR] Generic Onoff Server Initialization Failed. Result: 0x%04X\n",
        retval);
    }

    return retval;
}
```

Listing 6: Setting up as Provisionee Example

```
/* Provisionee */
#define UI_PROV_OUTPUT_OOB_ACTIONS \
    (PROV_MASK_OOOB_ACTION_BLINK | PROV_MASK_OOOB_ACTION_BEEP | \
     PROV_MASK_OOOB_ACTION_VIBRATE | PROV_MASK_OOOB_ACTION_NUMERIC | \
     PROV_MASK_OOOB_ACTION_ALPHANUMERIC)

/** Output OOB Maximum size supported */
#define UI_PROV_OUTPUT_OOB_SIZE                0x08

/** Input OOB Actions supported */
#define UI_PROV_INPUT_OOB_ACTIONS \
    (PROV_MASK_IOOB_ACTION_PUSH | PROV_MASK_IOOB_ACTION_TWIST | \
     PROV_MASK_IOOB_ACTION_NUMERIC | PROV_MASK_IOOB_ACTION_ALPHANUMERIC)

/** Input OOB Maximum size supported */
#define UI_PROV_INPUT_OOB_SIZE                 0x08

/** Beacon setup timeout in seconds */
#define UI_PROV_SETUP_TIMEOUT_SECS             30

/** Attention timeout for device in seconds */
#define UI_PROV_DEVICE_ATTENTION_TIMEOUT       30

#define PROV_AUTHVAL_SIZE_PL                   16

/** Authentication values for OOB Display - To be made random */
#define UI_DISPLAY_AUTH_DIGIT                  3
#define UI_DISPLAY_AUTH_NUMERIC                35007
#define UI_DISPLAY_AUTH_STRING                 "f001"

/** Provisioning capabilities of local device */
DECL_STATIC PROV_CAPABILITIES_S UI_prov_capab =
{
    /** Number of Elements */
    0x01,

    /** Supported algorithms */
    PROV_MASK_ALGO_EC_FIPS_P256,
```

Listing 6: Setting up as Provisionee Example

```
    /** Public key type */
    PROV_MASK_PUBKEY_OOBINFO,

    /** Static OOB type */
    PROV_MASK_STATIC_OOBINFO,

    /** Output OOB information */
    { UI_PROV_OUTPUT_OOB_ACTIONS, UI_PROV_OUTPUT_OOB_SIZE },

    /** Input OOB information */
    { UI_PROV_INPUT_OOB_ACTIONS, UI_PROV_INPUT_OOB_SIZE },
};

/** Unprovisioned device identifier */
DECL_STATIC PROV_DEVICE_S UI_lprov_device =
{
    /** UUID */
    {0x00, 0x11, 0x22, 0x33, 0x44, 0x55, 0x66, 0x77, 0x88, 0x99, 0xAA, 0xBB, 0xCC, 0xDD,
0xEE, 0xFF},

    /** OOB Flag */
    0x00,

    /**
     * Encoded URI Information
     * For example, to give a web address, "https://www.abc.com"
     * the URI encoded data would be -
     * 0x17 0x2F 0x2F 0x77 0x77 0x77 0x2E 0x61 0x62 0x63 0x2E 0x63 0x6F 0x6D
     * where 0x17 is the URI encoding for https:
     */
    NULL
};

/** Data exchanged during Provisiong procedure */
DECL_STATIC PROV_DATA_S UI_prov_data =
{
    /** NetKey */
    { 0x45, 0x74, 0x68, 0x65, 0x72, 0x4d, 0x69, 0x6e, 0x64, 0x4e, 0x65, 0x74, 0x4b, 0x65,
0x79, 0x00 },

    /** Index of the NetKey */
    0x0000,

    /** Flags bitmask */
    0x00,

    /** Current value of the IV index */
    0x00000001,

    /** Unicast address of the primary element */
    0x0002
};


/** Current role of application - Provisioner/Device */
DECL_STATIC UCHAR UI_prov_role;
```

Listing 6: Setting up as Provisionee Example

```
/** Provisioning Handle */
DECL_STATIC PROV_HANDLE UI_prov_handle;

API_RESULT UI_prov_callback
           (
               PROV_HANDLE * phandle,
               UCHAR         event_type,
               API_RESULT    event_result,
               void        * event_data,
               UINT16        event_datalen
           )
{
    PROV_DEVICE_S * rdev;
    PROV_CAPABILITIES_S * rcap;
    PROV_DATA_S * rdata;
    PROV_OOB_TYPE_S * oob_info;
    API_RESULT retval;
    UCHAR i;

    UCHAR authstr[PROV_AUTHVAL_SIZE_PL << 1];
    UINT32 authnum;
    UCHAR authtype;
    UCHAR * pauth;
    UINT16 authsize;
    UCHAR  pdata[(MS_DEVICE_UUID_SIZE * 2) + 1];
    UCHAR  * t_data;

    switch (event_type)
    {
        case PROV_EVT_PROVISIONING_SETUP:
            CONSOLE_OUT("Recvd PROV_EVT_PROVISIONING_SETUP\n");
            CONSOLE_OUT("Status - 0x%04X\n", event_result);

            /* Display the attention timeout */
            CONSOLE_OUT("Attention TImeout - %d\n", *((UCHAR *)event_data));
            break;

        case PROV_EVT_OOB_DISPLAY:
            CONSOLE_OUT("Recvd PROV_EVT_OOB_DISPLAY\n");
            CONSOLE_OUT("Status - 0x%04X\n", event_result);

            /* Reference the Authenticatio Type information */
            oob_info = (PROV_OOB_TYPE_S *)event_data;

            CONSOLE_OUT("Authenticaion Action - 0x%02X\n", oob_info->action);
            CONSOLE_OUT("Authenticaion Size - 0x%02X\n", oob_info->size);

            /* If role is Device, the action is of Output OOB, else Input OOB */
            if (PROV_ROLE_DEVICE == UI_prov_role)
            {
                if (PROV_OOOB_ACTION_ALPHANUMERIC == oob_info->action)
                {
                    authtype = 1;
                }
                else if (PROV_OOOB_ACTION_NUMERIC == oob_info->action)
                {
                    authtype = 2;
```

Listing 6: Setting up as Provisionee Example

```
            }
            else
            {
                authtype = 0;
            }
        }
        else
        {
            if (PROV_IOOB_ACTION_ALPHANUMERIC == oob_info->action)
            {
                authtype = 1;
            }
            else if (PROV_IOOB_ACTION_NUMERIC == oob_info->action)
            {
                authtype = 2;
            }
            else
            {
                authtype = 0;
            }
        }

        if (1 == authtype)
        {
            EM_str_copy (authstr, UI_DISPLAY_AUTH_STRING);

            CONSOLE_OUT("\n\n>>> AuthVal - %s <<<\n\n", authstr);

            pauth = authstr;
            authsize = EM_str_len(authstr);
        }
        else if (2 == authtype)
        {
            authnum = (UINT32)UI_DISPLAY_AUTH_NUMERIC;

            CONSOLE_OUT("\n\n>>> AuthVal - %d <<<\n\n", authnum);

            pauth = &authnum;
            authsize = sizeof(UINT32);
        }
        else
        {
            authnum = (UINT32)UI_DISPLAY_AUTH_DIGIT;

            CONSOLE_OUT("\n\n>>> AuthVal - %d <<<\n\n", authnum);

            pauth = &authnum;
            authsize = sizeof(UINT32);
        }

        /* Call to input the oob */
        CONSOLE_OUT("Setting the Authval...\n");
        retval = MS_prov_set_authval(&UI_prov_handle, pauth, authsize);
        CONSOLE_OUT("Retval - 0x%04X\n", retval);

        break;
```

Listing 6: Setting up as Provisionee Example

```
        case PROV_EVT_OOB_ENTRY:
            CONSOLE_OUT("Recvd PROV_EVT_OOB_ENTRY\n");
            CONSOLE_OUT("Status - 0x%04X\n", event_result);

            /* Reference the Authenticatio Type information */
            oob_info = (PROV_OOB_TYPE_S *)event_data;

            CONSOLE_OUT("Authenticaion Action - 0x%02X\n", oob_info->action);
            CONSOLE_OUT("Authenticaion Size - 0x%02X\n", oob_info->size);

            break;

        case PROV_EVT_DEVINPUT_COMPLETE:
            CONSOLE_OUT("Recvd PROV_EVT_DEVINPUT_COMPLETE\n");
            CONSOLE_OUT("Status - 0x%04X\n", event_result);

            break;

        case PROV_EVT_PROVDATA_INFO:
            CONSOLE_OUT("Recvd PROV_EVT_PROVDATA_INFO\n");
            CONSOLE_OUT("Status - 0x%04X\n", event_result);

            /* Reference the Provisioning Data */
            rdata = (PROV_DATA_S *)event_data;

            CONSOLE_OUT("NetKey  : "); appl_dump_bytes(rdata->netkey,
PROV_KEY_NETKEY_SIZE);
            CONSOLE_OUT("Key ID  : 0x%04X\n", rdata->keyid);
            CONSOLE_OUT("Flags   : 0x%02X\n", rdata->flags);
            CONSOLE_OUT("IVIndex : 0x%08X\n", rdata->ivindex);
            CONSOLE_OUT("UAddr   : 0x%04X\n", rdata->uaddr);

            /* Provide Provisioning Data to Access Layer */
            MS_access_cm_set_prov_data
            (
                 rdata
            );

            break;

        case PROV_EVT_PROVISIONING_COMPLETE:
            CONSOLE_OUT("Recvd PROV_EVT_PROVISIONING_COMPLETE\n");
            CONSOLE_OUT("Status - 0x%04X\n", event_result);

            if (API_SUCCESS == event_result)
            {
                /* Already Set while handling PROV_EVT_PROVDATA_INFO */
            }
            break;

        default:
            CONSOLE_OUT("Unknown Event - 0x%02X\n", event_type);
    }

    return API_SUCCESS;
}
```

Listing 6: Setting up as Provisionee Example

```
void UI_register_prov(void)
{
    API_RESULT retval;

    CONSOLE_OUT("Registering with Provisioning layer...\n");
    retval = MS_prov_register(&UI_prov_capab, UI_prov_callback);
    CONSOLE_OUT("Retval - 0x%04X\n", retval);
}

void UI_setup_prov(UCHAR role, UCHAR brr)
{
    API_RESULT retval;

    if (PROV_ROLE_PROVISIONER != role)
    {
        CONSOLE_OUT("Setting up Device for Provisioning ...\n");
        retval = MS_prov_setup
                 (
                     brr,
                     role,
                     &UI_lprov_device,
                     UI_PROV_SETUP_TIMEOUT_SECS
                 );

        UI_prov_role = PROV_ROLE_DEVICE;
    }
    else
    {
        CONSOLE_OUT("Setting up Provisioner for Provisioning ...\n");
        retval = MS_prov_setup
                 (
                     brr,
                     role,
                     NULL,
                     UI_PROV_SETUP_TIMEOUT_SECS
                 );

        UI_prov_role = PROV_ROLE_PROVISIONER;
    }

    CONSOLE_OUT("Retval - 0x%04X\n", retval);
}

void UI_prov_bind(UCHAR brr, UCHAR index)
{
    API_RESULT retval;

    /* Call to bind with the selected device */
    CONSOLE_OUT("Binding with the selected device...\n");

    retval = MS_prov_bind(brr, &UI_lprov_device, UI_PROV_DEVICE_ATTENTION_TIMEOUT,
&UI_prov_handle);

    CONSOLE_OUT("Retval - 0x%04X\n", retval);
}
```

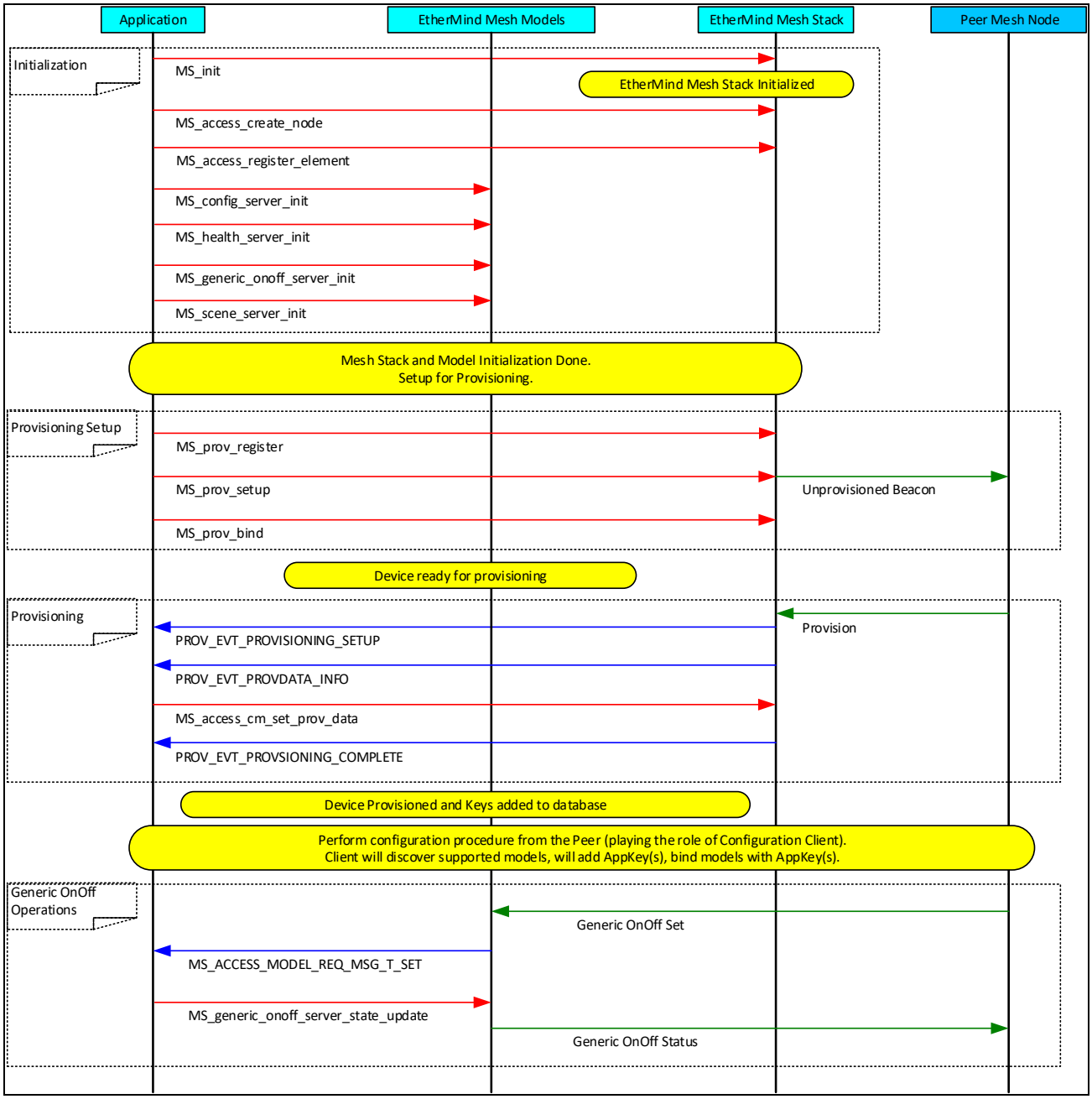
Listing 7: Putting all together Example

```
void UI_Main (param, ...)
{
    MS_CONFIG * config_ptr;
    MS_ACCESS_NODE_ID node_id;
    MS_ACCESS_ELEMENT_DESC   element;
    MS_ACCESS_ELEMENT_HANDLE element_handle;
    API_RESULT retval;

    UCHAR role, brr;

#ifdef MS_HAVE_DYNAMIC_CONFIG
    MS_CONFIG  config;

    /* Initialize dynamic configuration */
    MS_INIT_CONFIG(config);

    config_ptr = &config;
#else
    config_ptr = NULL;
#endif /* MS_HAVE_DYNAMIC_CONFIG */

    /* Initialize OSAL */
    EM_os_init();

    /* Initialize Debug Module */
    EM_debug_init();

    /* Initialize Timer Module */
    EM_timer_init();
    timer_em_init();

    /* Initialize utilities */
    nvsto_init();

    /* Initialize Mesh Stack */
    MS_init(config_ptr);

    /* Register with underlying BLE stack */
    blebrr_register();

    /* Create Node */
    retval = MS_access_create_node(&node_id);

    /* Register Element */
    /**
     * TBD: Define GATT Namespace Descriptions from
     * https://www.bluetooth.com/specifications/assigned-numbers/gatt-namespace-
descriptors
     *
     * Using 'main' (0x0106) as Location temporarily.
     */
    element.loc = 0x0106;

    retval = MS_access_register_element
             (
                 node_id,
                 &element,
```

Listing 7: Putting all together Example

```
                &element_handle
            );

    if (API_SUCCESS == retval)
    {
        /* Register foundation model servers */
        retval = UI_register_foundation_model_servers(element_handle);
    }

    if (API_SUCCESS == retval)
    {
        /* Register Generic OnOff model server */
        retval = UI_register_generic_onoff_model_server(element_handle);
    }

    if (API_SUCCESS == retval)
    {
        /* Initialize model states */
        UI_model_states_initialization();
    }

    /* Configure as provisionee/device */
    UI_register_prov();

    /**
     * setup <role:[1 - Device, 2 - Provisioner]> <bearer:[1 - Adv, 2 - GATT]
     */
    role = PROV_ROLE_DEVICE;
    brr = PROV_BRR_ADV;
    UI_setup_prov(role, brr);
    UI_prov_bind(brr, 0x00);

    /* Loop forever (if required) */
    MS_LOOP_FOREVER()
    {
        EM_sleep(10);
    }

    return;
}
```

Following diagram shows the usage of relevant APIs, putting all of these together:



How to add another model to the same element?

Light Lightness Model is used below as an example, to demonstrate how to add another model to the same element where Generic OnOff model is already added.

Most part of the application will be the same as Generic OnOff server application. Only a few parts need to be updated as shown below.

Listing 2-a: Header File Inclusion for Foundation and Additional Models

```
/* ---------------------------------------- Header File Inclusion */
#include "MS_common.h"
#include "MS_access_api.h"
#include "MS_config_api.h"
#include "MS_health_server_api.h"
#include "MS_generic_onoff_api.h"
#include "MS_light_lightness_api.h"

/* Console Input/Output */
#define CONSOLE_OUT(...)    printf(__VA_ARGS__)
#define CONSOLE_IN(...)     scanf(__VA_ARGS__)
```

Listing 4-a: Model Initialization, Get/Set state handlers and Callback routine Example

```
/* ---- Generic OnOff States */
static MS_STATE_GENERIC_ONOFF_STRUCT UI_generic_onoff;

/** -- Light - Lightness */
static MS_STATE_LIGHT_LIGHTNESS_STRUCT UI_light_lightness;

/* Get/Set State Handlers */

/* Generic OnOff Model state Initialization */
void UI_generic_onoff_model_states_initialization(void)
{
    EM_mem_set(&UI_generic_onoff, 0, sizeof(UI_generic_onoff));
}

/* Generic OnOff Model Get Handler */
API_RESULT UI_generic_onoff_model_state_get(UINT16 state_t, UINT16 state_inst, void *
param, UINT8 direction)
{
    API_RESULT retval;

    retval = API_SUCCESS;

    switch(state_t)
    {
        case MS_STATE_GENERIC_ONOFF_T:
        {
            MS_STATE_GENERIC_ONOFF_STRUCT * param_p;

            param_p = (MS_STATE_GENERIC_ONOFF_STRUCT *)param;

            /* Ignoring Instance and direction right now */
            *param_p = UI_generic_onoff;
        }
        break;

        default:
        break;
    }

    return retval;
}
```

Listing 4-a: Model Initialization, Get/Set state handlers and Callback routine Example

```
/* Generic OnOff Model Set Handler */
API_RESULT UI_generic_onoff_model_state_set(UINT16 state_t, UINT16 state_inst, void *
param, UINT8 direction)
{
    API_RESULT retval;

    retval = API_SUCCESS;
    switch (state_t)
    {
        case MS_STATE_GENERIC_ONOFF_T:
        {
            MS_STATE_GENERIC_ONOFF_STRUCT * param_p;

            param_p = (MS_STATE_GENERIC_ONOFF_STRUCT *)param;

            /* Instantaneous Change */
            UI_generic_onoff.onoff = param_p->onoff;

            *param_p = UI_generic_onoff;
            CONSOLE_OUT("[state] current: 0x%02X\n", UI_generic_onoff.onoff);
            CONSOLE_OUT("[state] target: 0x%02X\n", UI_generic_onoff.target_onoff);
            CONSOLE_OUT("[state] remaining_time: 0x%02X\n",
UI_generic_onoff.transition_time);

            /* Ignoring Instance and direction right now */
        }
        break;

        default:
        break;
    }

    return retval;
}

/* Light Lightness Model state Initialization */
void UI_light_lightness_model_states_initialization(void)
{
    /* Light Lightness States */
    EM_mem_set(&UI_light_lightness, 0, sizeof(UI_light_lightness));
    UI_light_lightness.light_lightness_last.lightness_last = 0xFFFF;
}

/* Light Lightness Model Get Handler */
API_RESULT UI_light_lightness_model_state_get(UINT16 state_t, UINT16 state_inst, void *
param, UINT8 direction)
{
    MS_STATE_LIGHT_LIGHTNESS_STRUCT * param_p;
    API_RESULT retval;

    param_p = (MS_STATE_LIGHT_LIGHTNESS_STRUCT *)param;
    retval = API_SUCCESS;

    switch(state_t)
    {
        case MS_STATE_LIGHT_LIGHTNESS_DEFAULT_T:
```

Listing 4-a: Model Initialization, Get/Set state handlers and Callback routine Example

```
        {
            /* Ignoring Instance and direction right now */
            param_p->light_lightness_default = UI_light_lightness.light_lightness_default;
        }
        break;

        case MS_STATE_LIGHT_LIGHTNESS_RANGE_T:
        {
            /* Ignoring Instance and direction right now */
            param_p->light_lightness_range = UI_light_lightness.light_lightness_range;
            param_p->range_status = 0x00;
        }
        break;

        case MS_STATE_LIGHT_LIGHTNESS_LINEAR_T:
        {
            /* Ignoring Instance and direction right now */
            param_p->light_lightness_linear = UI_light_lightness.light_lightness_linear;
        }
        break;

        case MS_STATE_LIGHT_LIGHTNESS_LAST_T:
        {
            /* Ignoring Instance and direction right now */
            param_p->light_lightness_last = UI_light_lightness.light_lightness_last;
        }
        break;

        case MS_STATE_LIGHT_LIGHTNESS_ACTUAL_T:
        {
            /* Ignoring Instance and direction right now */
            param_p->light_lightness_actual = UI_light_lightness.light_lightness_actual;
        }
        break;

        default:
        break;
    }

    return retval;
}

/* Light Lightness Model Set Handler */
/* Todo: Remove the dependency */
#include "math.h"

static void UI_light_lightness_set_actual(UINT16 state_inst, UINT16 actual)
{
    UINT16 min, max;

    /* Generic OnOff binding */
    min = UI_light_lightness.light_lightness_range.lightness_range_min;
    max = UI_light_lightness.light_lightness_range.lightness_range_max;

    if ((0 != min) && (actual < min))
    {
        actual = min;
```

Listing 4-a: Model Initialization, Get/Set state handlers and Callback routine Example

```
    }
    else if ((0 != max) && (actual > max))
    {
        actual = max;
    }

    /* If Lightness Actual is non-zero, save as Lightness Last */
    if (0x0000 != actual)
    {
        UI_light_lightness.light_lightness_last.lightness_last = actual;
    }

    UI_light_lightness.light_lightness_actual.lightness_actual = actual;

    /* Light Lightness Linear = ((Actual)^2) / 65535 */
    UI_light_lightness.light_lightness_linear.lightness_linear = ((actual * actual) +
65534) / 65535;
}

static void UI_light_lightness_set_linear(UINT16 state_inst, UINT16 linear)
{
    UINT16 actual;
    UINT32 mul_val;
    long double d;

    mul_val = linear * 65535;
    actual = (UINT16)sqrt(mul_val);

    /* Light Lightness actual = sqrt(Linear * 65535) */
    UI_light_lightness_set_actual(state_inst, actual);
}

API_RESULT UI_light_lightness_model_state_set(UINT16 state_t, UINT16 state_inst, void *
param, UINT8 direction)
{
    MS_STATE_LIGHT_LIGHTNESS_STRUCT * param_p;
    API_RESULT retval;

    param_p = (MS_STATE_LIGHT_LIGHTNESS_STRUCT *)param;
    retval = API_SUCCESS;

    switch (state_t)
    {
        case MS_STATE_LIGHT_LIGHTNESS_DEFAULT_T:
        {
            /* Ignoring Instance and direction right now */
            UI_light_lightness.light_lightness_default = param_p->light_lightness_default;
        }
        break;

        case MS_STATE_LIGHT_LIGHTNESS_RANGE_T:
        {
            /* Check range min and max */
            if (param_p->light_lightness_range.lightness_range_min > param_p-
>light_lightness_range.lightness_range_max)
            {
                /* TODO: add macro define */
```

Listing 4-a: Model Initialization, Get/Set state handlers and Callback routine Example

```
                /**
                 * Table 7.2:
                 * 0x00 - Success
                 * 0x01 - Cannot Set Range Min
                 * 0x02 - Cannot Set Range Max
                 */
                param_p->range_status = 0x01;
            }
            else
            {
                /* Ignoring Instance and direction right now */
                UI_light_lightness.light_lightness_range = param_p->light_lightness_range;
                param_p->range_status = 0x00;
            }
        }
        break;

        case MS_STATE_LIGHT_LIGHTNESS_LINEAR_T:
        {
            /* Instantaneous Change */
            UI_light_lightness_set_linear(0, param_p-
>light_lightness_linear.lightness_linear);

            *param_p = UI_light_lightness;
            CONSOLE_OUT("[state] current: 0x%02X\n", param_p-
>light_lightness_linear.lightness_linear);
            CONSOLE_OUT("[state] target: 0x%02X\n", param_p-
>light_lightness_linear.lightness_target);
            CONSOLE_OUT("[state] remaining_time: 0x%02X\n", param_p-
>light_lightness_linear.transition_time);

            /* Ignoring Instance and direction right now */
        }
        break;

        case MS_STATE_LIGHT_LIGHTNESS_LAST_T:
        {
            /* Ignoring Instance and direction right now */
            UI_light_lightness.light_lightness_last = param_p->light_lightness_last;
        }
        break;

        case MS_STATE_LIGHT_LIGHTNESS_ACTUAL_T:
        {
            /* Instantaneous Change */
            UI_light_lightness_set_actual(0, param_p-
>light_lightness_actual.lightness_actual);

            *param_p = UI_light_lightness;
            CONSOLE_OUT("[state] current: 0x%02X\n", param_p-
>light_lightness_actual.lightness_actual);
            CONSOLE_OUT("[state] target: 0x%02X\n", param_p-
>light_lightness_actual.lightness_target);
            CONSOLE_OUT("[state] remaining_time: 0x%02X\n", param_p-
>light_lightness_actual.transition_time);
        }
        break;
```

Listing 4-a: Model Initialization, Get/Set state handlers and Callback routine Example

```
        default:
        break;
    }

    return retval;
}

/* Model state Initialization */
void UI_model_states_initialization(void)
{
    /* Generic OnOff States */
    UI_generic_onoff_model_states_initialization();

    /* Light Lightness States */
    UI_light_lightness_model_states_initialization();
}
```

Listing 5-a: Light Lightness Initialization and Callback routine Example

```
/* Light Ligthness Model Server */
/**
 * \brief Server Application Asynchronous Notification Callback.
 *
 * \par Description
 * Light_Lightness server calls the registered callback to indicate events occurred to the
application.
 *
 * \param [in] ctx           Context of message received for a specific model instance.
 * \param [in] msg_raw       Uninterpreted/raw received message.
 * \param [in] req_type      Requested message type.
 * \param [in] state_params  Model specific state parameters.
 * \param [in] ext_params    Additional parameters.
 */
API_RESULT UI_light_lightness_server_cb
           (
               /* IN */ MS_ACCESS_MODEL_REQ_MSG_CONTEXT    * ctx,
               /* IN */ MS_ACCESS_MODEL_REQ_MSG_RAW        * msg_raw,
               /* IN */ MS_ACCESS_MODEL_REQ_MSG_T          * req_type,
               /* IN */ MS_ACCESS_MODEL_STATE_PARAMS       * state_params,
               /* IN */ MS_ACCESS_MODEL_EXT_PARAMS         * ext_params
           )
{
    MS_STATE_LIGHT_LIGHTNESS_STRUCT param;
    MS_ACCESS_MODEL_STATE_PARAMS                    current_state_params;

    API_RESULT retval;

    retval = API_SUCCESS;

    /* Check message type */
    if (MS_ACCESS_MODEL_REQ_MSG_T_GET == req_type->type)
    {
        CONSOLE_OUT(
        "[LIGHT_LIGHTNESS] GET Request.\n");
```

Listing 5-a: Light Lightness Initialization and Callback routine Example

```
        UI_light_lightness_model_state_get(state_params->state_type, 0, &param, 0);

        current_state_params.state_type = state_params->state_type;
        current_state_params.state = &param;
    }
    else if (MS_ACCESS_MODEL_REQ_MSG_T_SET == req_type->type)
    {
        CONSOLE_OUT(
        "[LIGHT_LIGHTNESS] SET Request.\n");

        UI_light_lightness_model_state_set(state_params->state_type, 0,
(MS_STATE_LIGHT_LIGHTNESS_STRUCT *)state_params->state, 0);

        current_state_params.state_type = state_params->state_type;
        current_state_params.state = (MS_STATE_LIGHT_LIGHTNESS_STRUCT *)state_params-
>state;
    }

    /* See if to be acknowledged */
    if (0x01 == req_type->to_be_acked)
    {
        CONSOLE_OUT(
        "[LIGHT_LIGHTNESS] Sending Response.\n");

        /* Parameters: Request Context, Current State, Target State (NULL: to be ignored),
Remaining Time (0: to be ignored), Additional Parameters (NULL: to be ignored) */
        retval = MS_light_lightness_server_state_update(ctx, &current_state_params, NULL,
0, NULL);
    }

    return retval;
}

API_RESULT UI_register_light_lightness_model_server
           (
               MS_ACCESS_ELEMENT_HANDLE element_handle
           )
{
    /* Generic OnOff Server */
    MS_ACCESS_MODEL_HANDLE   UI_light_lightness_server_model_handle;
    API_RESULT retval;

    retval = MS_light_lightness_server_init
             (
                 element_handle,
                 &UI_light_lightness_server_model_handle,
                 UI_light_lightness_server_cb
             );

    if (API_SUCCESS == retval)
    {
        CONSOLE_OUT(
        "Light Lightness Server Initialized. Model Handle: 0x%04X\n",
        UI_light_lightness_server_model_handle);
    }
    else
```

Listing 5-a: Light Lightness Initialization and Callback routine Example

```
    {
        CONSOLE_OUT(
        "[ERR] Light Lightness Server Initialization Failed. Result: 0x%04X\n",
        retval);
    }

    return retval;
}
```

Listing 7-a: Putting all together Example

```
void main (void)
{
    MS_CONFIG * config_ptr;
    MS_ACCESS_NODE_ID node_id;
    MS_ACCESS_ELEMENT_DESC   element;
    MS_ACCESS_ELEMENT_HANDLE element_handle;
    API_RESULT retval;

    UCHAR role, brr;

#ifdef MS_HAVE_DYNAMIC_CONFIG
    MS_CONFIG  config;

    /* Initialize dynamic configuration */
    MS_INIT_CONFIG(config);

    config_ptr = &config;
#else
    config_ptr = NULL;
#endif /* MS_HAVE_DYNAMIC_CONFIG */

    /* Initialize OSAL */
    EM_os_init();

    /* Initialize Debug Module */
    EM_debug_init();

    /* Initialize Timer Module */
    EM_timer_init();
    timer_em_init();

    /* Initialize utilities */
    nvsto_init();

    /* Initialize Mesh Stack */
    MS_init(config_ptr);

    /* Register with underlying BLE stack */
    blebrr_register();

    /* Create Node */
    retval = MS_access_create_node(&node_id);
```

Listing 7-a: Putting all together Example

```
    /* Register Element */
    /**
     * TBD: Define GATT Namespace Descriptions from
     * https://www.bluetooth.com/specifications/assigned-numbers/gatt-namespace-
descriptors
     *
     * Using 'main' (0x0106) as Location temporarily.
     */
    element.loc = 0x0106;

    retval = MS_access_register_element
             (
                 node_id,
                 &element,
                 &element_handle
             );

    if (API_SUCCESS == retval)
    {
        /* Register foundation model servers */
        retval = UI_register_foundation_model_servers(element_handle);
    }

    if (API_SUCCESS == retval)
    {
        /* Register Generic OnOff model server */
        retval = UI_register_generic_onoff_model_server(element_handle);
    }

    if (API_SUCCESS == retval)
    {
        /* Register Light Lightness model server */
        retval = UI_register_light_lightness_model_server(element_handle);
    }

    if (API_SUCCESS == retval)
    {
        /* Initialize model states */
        UI_model_states_initialization();
    }

    /* Configure as provisionee/device */
    UI_register_prov();

    /**
     * setup <role:[1 - Device, 2 - Provisioner]> <bearer:[1 - Adv, 2 - GATT]
     */
    role = PROV_ROLE_DEVICE;
    brr = PROV_BRR_ADV;
    UI_setup_prov(role, brr);
    UI_prov_bind(brr, 0x00);

    /* Loop forever */
    MS_LOOP_FOREVER()
    {
        EM_sleep(10);
```

Listing 7-a: Putting all together Example

```
    }

    return;
}
```

How to add another model to a different element?

Light Lightness Model is used below as an example, to demonstrate how to add another model to an element different from where Generic OnOff model is already added.

Most part of the application will be the same as Generic OnOff server and Light Lightness application. Only a few parts need to be updated as shown below.

Listing 6-a: Setting up as Provisionee Example - Number of elements updated

```
/*
 * All other defines remain same, except for changing Number of elements as 2,
 * in UI_prov_capab data structure
 */
...
...
/** Authentication values for OOB Display - To be made random */
#define UI_DISPLAY_AUTH_DIGIT                 3
#define UI_DISPLAY_AUTH_NUMERIC               35007
#define UI_DISPLAY_AUTH_STRING                "f001"

/** Provisioning capabilities of local device */
DECL_STATIC PROV_CAPABILITIES_S UI_prov_capab =
{
    /** Number of Elements */
    0x02,

    /** Supported algorithms */
    PROV_MASK_ALGO_EC_FIPS_P256,

    /** Public key type */
    PROV_MASK_PUBKEY_OOBINFO,

    /** Static OOB type */
    PROV_MASK_STATIC_OOBINFO,

    /** Output OOB information */
    { UI_PROV_OUTPUT_OOB_ACTIONS, UI_PROV_OUTPUT_OOB_SIZE },

    /** Input OOB information */
    { UI_PROV_INPUT_OOB_ACTIONS, UI_PROV_INPUT_OOB_SIZE },
};

/** Unprovisioned device identifier */
DECL_STATIC PROV_DEVICE_S UI_lprov_device =
{
    /** UUID */
    {0x00, 0x11, 0x22, 0x33, 0x44, 0x55, 0x66, 0x77, 0x88, 0x99, 0xAA, 0xBB, 0xCC, 0xDD,
0xEE, 0xFF},
```

Listing 6-a: Setting up as Provisionee Example - Number of elements updated

```
...
...
```

Listing 7-b: Putting all together Example

```
void main (void)
{
    MS_CONFIG * config_ptr;
    MS_ACCESS_NODE_ID node_id;
    MS_ACCESS_ELEMENT_DESC   element;
    MS_ACCESS_ELEMENT_HANDLE element_handle;

    MS_ACCESS_ELEMENT_DESC   element_1;
    MS_ACCESS_ELEMENT_HANDLE element_handle_1;
    API_RESULT retval;

    UCHAR role, brr;

#ifdef MS_HAVE_DYNAMIC_CONFIG
    MS_CONFIG  config;

    /* Initialize dynamic configuration */
    MS_INIT_CONFIG(config);

    config_ptr = &config;
#else
    config_ptr = NULL;
#endif /* MS_HAVE_DYNAMIC_CONFIG */

    /* Initialize OSAL */
    EM_os_init();

    /* Initialize Debug Module */
    EM_debug_init();

    /* Initialize Timer Module */
    EM_timer_init();
    timer_em_init();

    /* Initialize utilities */
    nvsto_init();

    /* Initialize Mesh Stack */
    MS_init(config_ptr);

    /* Register with underlying BLE stack */
    blebrr_register();

    /* Create Node */
    retval = MS_access_create_node(&node_id);

    /* Register Element */
    /**
     * TBD: Define GATT Namespace Descriptions from
     * https://www.bluetooth.com/specifications/assigned-numbers/gatt-namespace-
```

Listing 7-b: Putting all together Example

```
descriptors
     *
     * Using 'main' (0x0106) as Location temporarily.
     */
    element.loc = 0x0106;

    retval = MS_access_register_element
             (
                 node_id,
                 &element,
                 &element_handle
             );

    if (API_SUCCESS == retval)
    {
        /* Register foundation model servers */
        retval = UI_register_foundation_model_servers(element_handle);
    }

    if (API_SUCCESS == retval)
    {
        /* Register Generic OnOff model server */
        retval = UI_register_generic_onoff_model_server(element_handle);
    }

    /* Register another Element */
    /**
     * TBD: Define GATT Namespace Descriptions from
     * https://www.bluetooth.com/specifications/assigned-numbers/gatt-namespace-
descriptors
     *
     * Using 'main' (0x0106) as Location temporarily.
     */
    element_1.loc = 0x0106;

    retval = MS_access_register_element
             (
                 node_id,
                 &element_1,
                 &element_handle_1
             );

    if (API_SUCCESS == retval)
    {
        /* Register Light Lightness model server */
        retval = UI_register_light_lightness_model_server(element_handle_1);
    }

    if (API_SUCCESS == retval)
    {
        /* Initialize model states */
        UI_model_states_initialization();
    }

    /* Configure as provisionee/device */
    UI_register_prov();
```

Listing 7-b: Putting all together Example

```
    /**
     * setup <role:[1 - Device, 2 - Provisioner]> <bearer:[1 - Adv, 2 - GATT]
     */
    role = PROV_ROLE_DEVICE;
    brr = PROV_BRR_ADV;
    UI_setup_prov(role, brr);
    UI_prov_bind(brr, 0x00);

    /* Loop forever */
    MS_LOOP_FOREVER()
    {
        EM_sleep(10);
    }

    return;
}
```

How to write a Model client application?

Generic OnOff Model client is used below as an example to explore the usage of Mesh APIs.

Generic OnOff Client Application

Generic OnOff Client uses following set of APIs

[1] Initialization of Core Mesh Stack

[2] Initialization of Configuration Client

[3] Initializations of Generic OnOff Client

[4] Provision APIs for provisioner

Generic OnOff Client example for Advertising Bearer.

Note

Steps [1], [2] and [4] will be the same for all model client implementation.

Listing 2-a: Header File Inclusion for Configuration and Generic OnOff Models

```
/* ---------------------------------------- Header File Inclusion */
#include "MS_common.h"
#include "MS_access_api.h"
#include "MS_config_api.h"
#include "MS_generic_onoff_api.h"

/* Console Input/Output */
#define CONSOLE_OUT(...)    printf(__VA_ARGS__)
#define CONSOLE_IN(...)     scanf(__VA_ARGS__)
```

✍ **Note**

✍ It is not mandatory to support the foundation server models for an implementation playing the role of a Provisioner and Configuration Client.

Listing 3-a: Foundation (Configuration) Model Client Initialization and helper routines Example

```
/* Configuration Client Model Handle */
DECL_STATIC MS_ACCESS_MODEL_HANDLE   UI_config_client_model_handle;

/** Appkey to be used for model binding */
DECL_STATIC UCHAR UI_appkey[MS_ACCESS_APPKEY_SIZE] = UI_APPKEY;

/* Model Client - Configuration Models */
/* Send Config Composition Data Get */
void UI_config_client_get_composition_data(UCHAR page)
{
    API_RESULT retval;
    ACCESS_CONFIG_COMPDATA_GET_PARAM  param;

    CONSOLE_OUT
    ("Send Config Composition Data Get\n");

    param.page = page;
    retval = MS_config_client_composition_data_get(&param);

    CONSOLE_OUT
    ("Retval - 0x%04X\n", retval);
}

/* Send Config Appkey Add */
void UI_config_client_appkey_add(UINT16 netkey_index, UINT16 appkey_index, UCHAR * appkey)
{
    API_RESULT retval;
    ACCESS_CONFIG_APPKEY_ADD_PARAM  param;

    CONSOLE_OUT
    ("Send Config Appkey Add\n");

    param.netkey_index = netkey_index;
    param.appkey_index = appkey_index;
    EM_mem_copy(param.appkey, appkey, MS_ACCESS_APPKEY_SIZE);

    /* Update local database */
    MS_access_cm_add_appkey
    (
        0, /* subnet_handle */
        param.appkey_index, /* appkey_index */
        &param.appkey[0] /* app_key */
    );

    retval = MS_config_client_appkey_add(&param);

    CONSOLE_OUT
    ("Retval - 0x%04X\n", retval);
}
```

Listing 3-a: Foundation (Configuration) Model Client Initialization and helper routines Example

```
void UI_config_client_model_app_bind(UINT16 addr, UINT16 appkey_index, UCHAR model_type,
UINT32 model_id)
{
    API_RESULT retval;
    ACCESS_CONFIG_MODEL_APP_BIND_PARAM  param;

    CONSOLE_OUT
    ("Send Config Model App Bind\n");

    param.element_address = addr;
    param.appkey_index = appkey_index;
    param.model.type = model_type;
    param.model.id = model_id;
    retval = MS_config_client_model_app_bind(&param);

    CONSOLE_OUT
    ("Retval - 0x%04X\n", retval);
}

/* Set Publish Address */
void UI_set_publish_address(UINT16 addr, MS_ACCESS_MODEL_HANDLE model_handle)
{
    API_RESULT retval;
    MS_ACCESS_PUBLISH_INFO  publish_info;

    /* Set Publish Information */
    EM_mem_set(&publish_info, 0, sizeof(publish_info));

    publish_info.addr.use_label = MS_FALSE;
    publish_info.appkey_index = MS_CONFIG_LIMITS(MS_MAX_APPS);
    publish_info.remote = MS_FALSE;
    publish_info.addr.addr = addr;

    retval = MS_access_cm_set_model_publication
             (
                 model_handle,
                 &publish_info
             );

    if (API_SUCCESS == retval)
    {
        CONSOLE_OUT
        ("Publish Address is set Successfully.\n");
    }
    else
    {
        CONSOLE_OUT
        ("Failed to set publish address. Status 0x%04X\n", retval);
    }

    return;
}

/**
 * \brief Client Application Asynchronous Notification Callback.
 *
 * \par Description
```

Listing 3-a: Foundation (Configuration) Model Client Initialization and helper routines Example

```
 * Configuration client calls the registered callback to indicate events occurred to the
application.
 *
 * \param [in] handle        Model Handle.
 * \param [in] opcode        Opcode.
 * \param [in] data_param    Data associated with the event if any or NULL.
 * \param [in] data_len      Size of the event data. 0 if event data is NULL.
 */
API_RESULT UI_config_client_cb
           (
                /* IN */ MS_ACCESS_MODEL_HANDLE * handle,
                /* IN */ UINT32                   opcode,
                /* IN */ UCHAR                  * data_param,
                /* IN */ UINT16                   data_len
           )
{
    API_RESULT retval;

    retval = API_SUCCESS;

    CONSOLE_OUT (
    "[CONFIG_CLIENT] Callback. Opcode 0x%04X\n", opcode);

    appl_dump_bytes(data_param, data_len);

    switch(opcode)
    {
        case MS_ACCESS_CONFIG_GATT_PROXY_STATUS_OPCODE:
        {
            CONSOLE_OUT(
            "MS_ACCESS_CONFIG_GATT_PROXY_STATUS_OPCODE\n");
        }
        break;

        case MS_ACCESS_CONFIG_HEARTBEAT_PUBLICATION_STATUS_OPCODE:
        {
            CONSOLE_OUT(
            "MS_ACCESS_CONFIG_HEARTBEAT_PUBLICATION_STATUS_OPCODE\n");
        }
        break;

        case MS_ACCESS_CONFIG_SIG_MODEL_APP_LIST_OPCODE:
        {
            CONSOLE_OUT(
            "MS_ACCESS_CONFIG_SIG_MODEL_APP_LIST_OPCODE\n");
        }
        break;

        case MS_ACCESS_CONFIG_COMPOSITION_DATA_STATUS_OPCODE:
        {
            CONSOLE_OUT(
            "MS_ACCESS_CONFIG_COMPOSITION_DATA_STATUS_OPCODE\n");

            /* Add Appkey */
            UI_config_client_appkey_add(0, 0, UI_appkey);
        }
        break;
```

Listing 3-a: Foundation (Configuration) Model Client Initialization and helper routines Example

```
        case MS_ACCESS_CONFIG_MODEL_SUBSCRIPTION_STATUS_OPCODE:
        {
            CONSOLE_OUT(
            "MS_ACCESS_CONFIG_MODEL_SUBSCRIPTION_STATUS_OPCODE\n");
        }
        break;

        case MS_ACCESS_CONFIG_NETKEY_LIST_OPCODE:
        {
            CONSOLE_OUT(
            "MS_ACCESS_CONFIG_NETKEY_LIST_OPCODE\n");
        }
        break;

        case MS_ACCESS_CONFIG_LOW_POWER_NODE_POLLTIMEOUT_STATUS_OPCODE:
        {
            CONSOLE_OUT(
            "MS_ACCESS_CONFIG_LOW_POWER_NODE_POLLTIMEOUT_STATUS_OPCODE\n");
        }
        break;

        case MS_ACCESS_CONFIG_SIG_MODEL_SUBSCRIPTION_LIST_OPCODE:
        {
            CONSOLE_OUT(
            "MS_ACCESS_CONFIG_SIG_MODEL_SUBSCRIPTION_LIST_OPCODE\n");
        }
        break;

        case MS_ACCESS_CONFIG_VENDOR_MODEL_SUBSCRIPTION_LIST_OPCODE:
        {
            CONSOLE_OUT(
            "MS_ACCESS_CONFIG_VENDOR_MODEL_SUBSCRIPTION_LIST_OPCODE\n");
        }
        break;

        case MS_ACCESS_CONFIG_APPKEY_LIST_OPCODE:
        {
            CONSOLE_OUT(
            "MS_ACCESS_CONFIG_APPKEY_LIST_OPCODE\n");
        }
        break;

        case MS_ACCESS_CONFIG_APPKEY_STATUS_OPCODE:
        {
            CONSOLE_OUT(
            "MS_ACCESS_CONFIG_APPKEY_STATUS_OPCODE\n");

            /* Bind the model to Appkey */
            UI_config_client_model_app_bind(UI_prov_data.uaddr, 0,
MS_ACCESS_MODEL_TYPE_SIG, MS_MODEL_ID_GENERIC_ONOFF_SERVER);
        }
        break;

        case MS_ACCESS_CONFIG_RELAY_STATUS_OPCODE:
        {
            CONSOLE_OUT(
```

Listing 3-a: Foundation (Configuration) Model Client Initialization and helper routines Example

```
            "MS_ACCESS_CONFIG_RELAY_STATUS_OPCODE\n");
        }
        break;

        case MS_ACCESS_CONFIG_FRIEND_STATUS_OPCODE:
        {
            CONSOLE_OUT(
            "MS_ACCESS_CONFIG_FRIEND_STATUS_OPCODE\n");
        }
        break;

        case MS_ACCESS_CONFIG_MODEL_APP_STATUS_OPCODE:
        {
            CONSOLE_OUT(
            "MS_ACCESS_CONFIG_MODEL_APP_STATUS_OPCODE\n");

            /* Set the Publish address for Config Client */
            UI_set_publish_address(UI_prov_data.uaddr,
UI_generic_onoff_client_model_handle);

            /* Send a Generic ON */
        }
        break;

        case MS_ACCESS_CONFIG_NODE_RESET_STATUS_OPCODE:
        {
            CONSOLE_OUT(
            "MS_ACCESS_CONFIG_NODE_RESET_STATUS_OPCODE\n");
        }
        break;

        case MS_ACCESS_CONFIG_DEFAULT_TTL_STATUS_OPCODE:
        {
            CONSOLE_OUT(
            "MS_ACCESS_CONFIG_DEFAULT_TTL_STATUS_OPCODE\n");
        }
        break;

        case MS_ACCESS_CONFIG_NODE_IDENTITY_STATUS_OPCODE:
        {
            CONSOLE_OUT(
            "MS_ACCESS_CONFIG_NODE_IDENTITY_STATUS_OPCODE\n");
        }
        break;

        case MS_ACCESS_CONFIG_HEARTBEAT_SUBSCRIPTION_STATUS_OPCODE:
        {
            CONSOLE_OUT(
            "MS_ACCESS_CONFIG_HEARTBEAT_SUBSCRIPTION_STATUS_OPCODE\n");
        }
        break;

        case MS_ACCESS_CONFIG_KEY_REFRESH_PHASE_STATUS_OPCODE:
        {
            CONSOLE_OUT(
            "MS_ACCESS_CONFIG_KEY_REFRESH_PHASE_STATUS_OPCODE\n");
        }
```

Listing 3-a: Foundation (Configuration) Model Client Initialization and helper routines Example

```
        break;

        case MS_ACCESS_CONFIG_NETWORK_TRANSMIT_STATUS_OPCODE:
        {
            CONSOLE_OUT(
            "MS_ACCESS_CONFIG_NETWORK_TRANSMIT_STATUS_OPCODE\n");
        }
        break;

        case MS_ACCESS_CONFIG_BEACON_STATUS_OPCODE:
        {
            CONSOLE_OUT(
            "MS_ACCESS_CONFIG_BEACON_STATUS_OPCODE\n");
        }
        break;

        case MS_ACCESS_CONFIG_MODEL_PUBLICATION_STATUS_OPCODE:
        {
            CONSOLE_OUT(
            "MS_ACCESS_CONFIG_MODEL_PUBLICATION_STATUS_OPCODE\n");
        }
        break;

        case MS_ACCESS_CONFIG_NETKEY_STATUS_OPCODE:
        {
            CONSOLE_OUT(
            "MS_ACCESS_CONFIG_NETKEY_STATUS_OPCODE\n");
        }
        break;

        case MS_ACCESS_CONFIG_VENDOR_MODEL_APP_LIST_OPCODE:
        {
            CONSOLE_OUT(
            "MS_ACCESS_CONFIG_VENDOR_MODEL_APP_LIST_OPCODE\n");
        }
        break;

    }

    return retval;
}

API_RESULT UI_register_config_model_client
           (
               MS_ACCESS_ELEMENT_HANDLE element_handle
           )
{
    /* Configuration Client */
    API_RESULT retval;

    CONSOLE_OUT("In Model Client - Configuration Models\n");

    retval = MS_config_client_init
             (
                 element_handle,
                 &UI_config_client_model_handle,
                 UI_config_client_cb
```

Listing 3-a: Foundation (Configuration) Model Client Initialization and helper routines Example

```
            );

    CONSOLE_OUT("Config Model Client Registration Status: 0x%04X\n", retval);

    return retval;
}
```

Listing 4-b: Generic OnOff Model client Initialization and helper routines Example

```
/* ---- Generic OnOff Handlers */
/* Generic ONOFF Client Model Handle */
DECL_STATIC MS_ACCESS_MODEL_HANDLE   UI_generic_onoff_client_model_handle;

void UI_generic_onoff_set(UCHAR state)
{
    API_RESULT retval;
    MS_GENERIC_ONOFF_SET_STRUCT  param;

    CONSOLE_OUT
    ("Send Generic Onoff Set\n");

    param.onoff = state;
    param.tid = 0;
    param.optional_fields_present = 0x00;
    retval = MS_generic_onoff_set(&param);

    CONSOLE_OUT
    ("Retval - 0x%04X\n", retval);
}


/* Generic OnOff Model Client */
/**
 * \brief Client Application Asynchronous Notification Callback.
 *
 * \par Description
 * Generic_Onoff client calls the registered callback to indicate events occurred to the
application.
 *
 * \param [in] handle        Model Handle.
 * \param [in] opcode        Opcode.
 * \param [in] data_param    Data associated with the event if any or NULL.
 * \param [in] data_len      Size of the event data. 0 if event data is NULL.
 */
API_RESULT UI_generic_onoff_client_cb
           (
                /* IN */ MS_ACCESS_MODEL_HANDLE * handle,
                /* IN */ UINT32                   opcode,
                /* IN */ UCHAR                  * data_param,
                /* IN */ UINT16                   data_len
           )
{
    API_RESULT retval;

    retval = API_SUCCESS;
```

Listing 4-b: Generic OnOff Model client Initialization and helper routines Example

```
    CONSOLE_OUT (
    "[GENERIC_ONOFF_CLIENT] Callback. Opcode 0x%04X\n", opcode);

    appl_dump_bytes(data_param, data_len);

    switch(opcode)
    {
        case MS_ACCESS_GENERIC_ONOFF_STATUS_OPCODE:
        {
            CONSOLE_OUT(
            "MS_ACCESS_GENERIC_ONOFF_STATUS_OPCODE\n");
        }
        break;

    }

    return retval;
}


API_RESULT UI_register_generic_onoff_model_client
           (
               MS_ACCESS_ELEMENT_HANDLE element_handle
           )
{
    /* Generic OnOff Client */
    API_RESULT retval;

    CONSOLE_OUT("In Generic OnOff Model Client\n");

    retval = MS_generic_onoff_client_init
             (
                 element_handle,
                 &UI_generic_onoff_client_model_handle,
                 UI_generic_onoff_client_cb
             );

    if (API_SUCCESS == retval)
    {
        CONSOLE_OUT(
        "Generic Onoff Client Initialized. Model Handle: 0x%04X\n",
        UI_generic_onoff_client_model_handle);
    }
    else
    {
        CONSOLE_OUT(
        "[ERR] Generic Onoff Client Initialization Failed. Result: 0x%04X\n",
        retval);
    }

    return retval;
}
```

Listing 6-b: Setting up as Provisioner Example

Listing 6-b: Setting up as Provisioner Example

```
/* Provisioner */
#define UI_PROV_OUTPUT_OOB_ACTIONS \
    (PROV_MASK_OOOB_ACTION_BLINK | PROV_MASK_OOOB_ACTION_BEEP | \
     PROV_MASK_OOOB_ACTION_VIBRATE | PROV_MASK_OOOB_ACTION_NUMERIC | \
     PROV_MASK_OOOB_ACTION_ALPHANUMERIC)

/** Output OOB Maximum size supported */
#define UI_PROV_OUTPUT_OOB_SIZE                0x08

/** Input OOB Actions supported */
#define UI_PROV_INPUT_OOB_ACTIONS \
    (PROV_MASK_IOOB_ACTION_PUSH | PROV_MASK_IOOB_ACTION_TWIST | \
     PROV_MASK_IOOB_ACTION_NUMERIC | PROV_MASK_IOOB_ACTION_ALPHANUMERIC)

/** Input OOB Maximum size supported */
#define UI_PROV_INPUT_OOB_SIZE                 0x08

/** Beacon setup timeout in seconds */
#define UI_PROV_SETUP_TIMEOUT_SECS             30

/** Attention timeout for device in seconds */
#define UI_PROV_DEVICE_ATTENTION_TIMEOUT       30

#define PROV_AUTHVAL_SIZE_PL                   16

/** Authentication values for OOB Display - To be made random */
#define UI_DISPLAY_AUTH_DIGIT                  3
#define UI_DISPLAY_AUTH_NUMERIC                35007
#define UI_DISPLAY_AUTH_STRING                 "f00l"

#define UI_DEVICE_UUID      {0x00, 0x11, 0x22, 0x33, 0x44, 0x55, 0x66, 0x77, 0x88, 0x99,
0xAA, 0xBB, 0xCC, 0xDD, 0xEE, 0xFF}

/** Provisioning capabilities of local device */
DECL_STATIC PROV_CAPABILITIES_S UI_prov_capab =
{
    /** Number of Elements */
    0x01,

    /** Supported algorithms */
    PROV_MASK_ALGO_EC_FIPS_P256,

    /** Public key type */
    PROV_MASK_PUBKEY_OOBINFO,

    /** Static OOB type */
    PROV_MASK_STATIC_OOBINFO,

    /** Output OOB information */
    { UI_PROV_OUTPUT_OOB_ACTIONS, UI_PROV_OUTPUT_OOB_SIZE },

    /** Input OOB information */
    { UI_PROV_INPUT_OOB_ACTIONS, UI_PROV_INPUT_OOB_SIZE },
};

/** Data exchanged during Provisiong procedure */
DECL_STATIC PROV_DATA_S UI_prov_data =
```

Listing 6-b: Setting up as Provisioner Example

```
{
    /** NetKey */
    { 0x45, 0x74, 0x68, 0x65, 0x72, 0x4d, 0x69, 0x6e, 0x64, 0x4e, 0x65, 0x74, 0x4b, 0x65,
0x79, 0x00 },

    /** Index of the NetKey */
    0x0000,

    /** Flags bitmask */
    0x00,

    /** Current value of the IV index */
    0x00000000,

    /** Unicast address of the primary element */
    0x0002
};

/** Default Provisioning method to start */
DECL_STATIC PROV_METHOD_S UI_prov_method =
{
    /** Algorithm */
    PROV_ALGO_EC_FIPS_P256,

    /** Public Key */
    PROV_PUBKEY_NO_OOB,

    /** Authentication Method */
    PROV_AUTH_OOB_NONE,

    /** OOB Information */
    {0x0000, 0x00}
};

/** Current role of application - Provisioner/Device */
DECL_STATIC UCHAR UI_prov_role;

/** Provisioning Handle */
DECL_STATIC PROV_HANDLE UI_prov_handle;

/** Device UUID Identifier */
DECL_STATIC UCHAR UI_device_uuid[MS_DEVICE_UUID_SIZE] = UI_DEVICE_UUID;

/* Set provision started */
DECL_STATIC UCHAR UI_prov_started;

API_RESULT UI_prov_callback
           (
               PROV_HANDLE * phandle,
               UCHAR         event_type,
               API_RESULT    event_result,
               void        * event_data,
               UINT16        event_datalen
           )
{
    PROV_DEVICE_S * rdev;
    PROV_CAPABILITIES_S * rcap;
```

Listing 6-b: Setting up as Provisioner Example

```
    PROV_DATA_S * rdata;
    PROV_OOB_TYPE_S * oob_info;
    API_RESULT retval;
    UCHAR i;

    UCHAR authstr[PROV_AUTHVAL_SIZE_PL << 1];
    UINT32 authnum;
    UCHAR authtype;
    UCHAR * pauth;
    UINT16 authsize;
    UCHAR  pdata[(MS_DEVICE_UUID_SIZE * 2) + 1];
    UCHAR  * t_data;

    switch (event_type)
    {
        case PROV_EVT_UNPROVISIONED_BEACON:

            /* Reference the beacon pointer */
            rdev = (PROV_DEVICE_S *)event_data;

            retval = API_SUCCESS;

            if (0 != EM_mem_cmp(rdev->uuid, UI_device_uuid, 16))
            {
                /* Beacon not from device of interest. Do no process */
                break;
            }

            CONSOLE_OUT ("Recvd PROV_EVT_UNPROVISIONED_BEACON\n");
            CONSOLE_OUT ("Status - 0x%04X\n", event_result);

            CONSOLE_OUT("\nUUID : [");

            EM_mem_set(pdata, 0x0, sizeof(pdata));
            t_data = pdata;

            for (i = 0; i < (MS_DEVICE_UUID_SIZE); i++)
            {
                sprintf(t_data,"%02X", rdev->uuid[i]);
                t_data += 2;
            }
            CONSOLE_OUT(" %s ", pdata);

            CONSOLE_OUT("]");

            CONSOLE_OUT("\nOOB  : 0x%04X", rdev->oob);
            CONSOLE_OUT("\nURI  : 0x%08X", rdev->uri);
            CONSOLE_OUT("\n\n");

            /* Bind to the device */
            retval = MS_prov_bind(PROV_BRR_ADV, rdev, UI_PROV_DEVICE_ATTENTION_TIMEOUT,
phandle);
            CONSOLE_OUT("Retval - 0x%04X\n", retval);

            /* Set provision started */
            UI_prov_started = MS_TRUE;
```

Listing 6-b: Setting up as Provisioner Example

```
            break;

        case PROV_EVT_PROVISIONING_SETUP:
            CONSOLE_OUT("Recvd PROV_EVT_PROVISIONING_SETUP\n");
            CONSOLE_OUT("Status - 0x%04X\n", event_result);

            /* Decipher the data based on the role */
            if (PROV_ROLE_PROVISIONER == UI_prov_role)
            {
                /* Display the capabilities */
                rcap = (PROV_CAPABILITIES_S *)event_data;

                CONSOLE_OUT ("Remote Device Capabilities:\n");
                CONSOLE_OUT ("\tNum Elements      - %d\n", rcap->num_elements);
                CONSOLE_OUT ("\tSupp Algorithms   - 0x%04X\n", rcap->supp_algorithms);
                CONSOLE_OUT ("\tSupp PublicKey    - 0x%02X\n", rcap->supp_pubkey);
                CONSOLE_OUT ("\tSupp Static OOB   - 0x%02X\n", rcap->supp_soob);
                CONSOLE_OUT ("\tOutput OOB Size   - %d\n", rcap->ooob.size);
                CONSOLE_OUT ("\tOutput OOB Action- 0x%04X\n", rcap->ooob.action);
                CONSOLE_OUT ("\tInput OOB Size    - %d\n", rcap->ioob.size);
                CONSOLE_OUT ("\tInput OOB Action - 0x%04X\n", rcap->ioob.action);

                /* Start with default method */
                CONSOLE_OUT ("Start Provisioning with default method...\n");
                retval = MS_prov_start (phandle, &UI_prov_method);
                CONSOLE_OUT ("Retval - 0x%04X\n", retval);
            }
            else
            {
                /* Display the attention timeout */
                CONSOLE_OUT("Attention TImeout - %d\n", *((UCHAR *)event_data));
            }
            break;

        case PROV_EVT_OOB_DISPLAY:
            CONSOLE_OUT("Recvd PROV_EVT_OOB_DISPLAY\n");
            CONSOLE_OUT("Status - 0x%04X\n", event_result);

            /* Reference the Authenticatio Type information */
            oob_info = (PROV_OOB_TYPE_S *)event_data;

            CONSOLE_OUT("Authenticaion Action - 0x%02X\n", oob_info->action);
            CONSOLE_OUT("Authenticaion Size - 0x%02X\n", oob_info->size);

            /* If role is Device, the action is of Output OOB, else Input OOB */
            if (PROV_ROLE_DEVICE == UI_prov_role)
            {
                if (PROV_OOOB_ACTION_ALPHANUMERIC == oob_info->action)
                {
                    authtype = 1;
                }
                else if (PROV_OOOB_ACTION_NUMERIC == oob_info->action)
                {
                    authtype = 2;
                }
                else
                {
```

Listing 6-b: Setting up as Provisioner Example

```
                    authtype = 0;
                }
            }
            else
            {
                if (PROV_IOOB_ACTION_ALPHANUMERIC == oob_info->action)
                {
                    authtype = 1;
                }
                else if (PROV_IOOB_ACTION_NUMERIC == oob_info->action)
                {
                    authtype = 2;
                }
                else
                {
                    authtype = 0;
                }
            }

            if (1 == authtype)
            {
                EM_str_copy (authstr, UI_DISPLAY_AUTH_STRING);

                CONSOLE_OUT("\n\n>>> AuthVal - %s <<<\n\n", authstr);

                pauth = authstr;
                authsize = EM_str_len(authstr);
            }
            else if (2 == authtype)
            {
                authnum = (UINT32)UI_DISPLAY_AUTH_NUMERIC;

                CONSOLE_OUT("\n\n>>> AuthVal - %d <<<\n\n", authnum);

                pauth = &authnum;
                authsize = sizeof(UINT32);
            }
            else
            {
                authnum = (UINT32)UI_DISPLAY_AUTH_DIGIT;

                CONSOLE_OUT("\n\n>>> AuthVal - %d <<<\n\n", authnum);

                pauth = &authnum;
                authsize = sizeof(UINT32);
            }

            /* Call to input the oob */
            CONSOLE_OUT("Setting the Authval...\n");
            retval = MS_prov_set_authval(phandle, pauth, authsize);
            CONSOLE_OUT("Retval - 0x%04X\n", retval);
            break;

        case PROV_EVT_OOB_ENTRY:
            CONSOLE_OUT("Recvd PROV_EVT_OOB_ENTRY\n");
            CONSOLE_OUT("Status - 0x%04X\n", event_result);
```

Listing 6-b: Setting up as Provisioner Example

```
            /* Reference the Authenticatio Type information */
            oob_info = (PROV_OOB_TYPE_S *)event_data;

            CONSOLE_OUT("Authenticaion Action - 0x%02X\n", oob_info->action);
            CONSOLE_OUT("Authenticaion Size - 0x%02X\n", oob_info->size);
            break;

        case PROV_EVT_DEVINPUT_COMPLETE:
            CONSOLE_OUT("Recvd PROV_EVT_DEVINPUT_COMPLETE\n");
            CONSOLE_OUT("Status - 0x%04X\n", event_result);
            break;

        case PROV_EVT_PROVDATA_INFO_REQ:
            CONSOLE_OUT ("Recvd PROV_EVT_PROVDATA_INFO_REQ\n");
            CONSOLE_OUT ("Status - 0x%04X\n", event_result);

            /* Send Provisioning Data */
            CONSOLE_OUT ("Send Provisioning Data...\n");
            retval = MS_prov_data (phandle, &UI_prov_data);
            CONSOLE_OUT ("Retval - 0x%04X\n", retval);
            break;

        case PROV_EVT_PROVISIONING_COMPLETE:
            CONSOLE_OUT("Recvd PROV_EVT_PROVISIONING_COMPLETE\n");
            CONSOLE_OUT("Status - 0x%04X\n", event_result);

            if (API_SUCCESS == event_result)
            {
                if (PROV_ROLE_PROVISIONER == UI_prov_role)
                {
                    /* Set provision started */
                    UI_prov_started = MS_FALSE;

                    if (0x0002 == UI_prov_data.uaddr)
                    {
                        /* Holding a temporary structure for local prov data */
                        PROV_DATA_S temp_ps_prov_data;

                        EM_mem_copy
                        (
                            &temp_ps_prov_data,
                            &UI_prov_data,
                            sizeof(UI_prov_data)
                        );

                        /**
                         * Assigning the Local Unicast Address of the Provisioner
                         * to the Access Layer along with other related keys.
                         */
                        temp_ps_prov_data.uaddr = 0x0001;

                        /* Provide Provisioning Data to Access Layer */
                        MS_access_cm_set_prov_data
                        (
                            &temp_ps_prov_data
                        );
```

Listing 6-b: Setting up as Provisioner Example

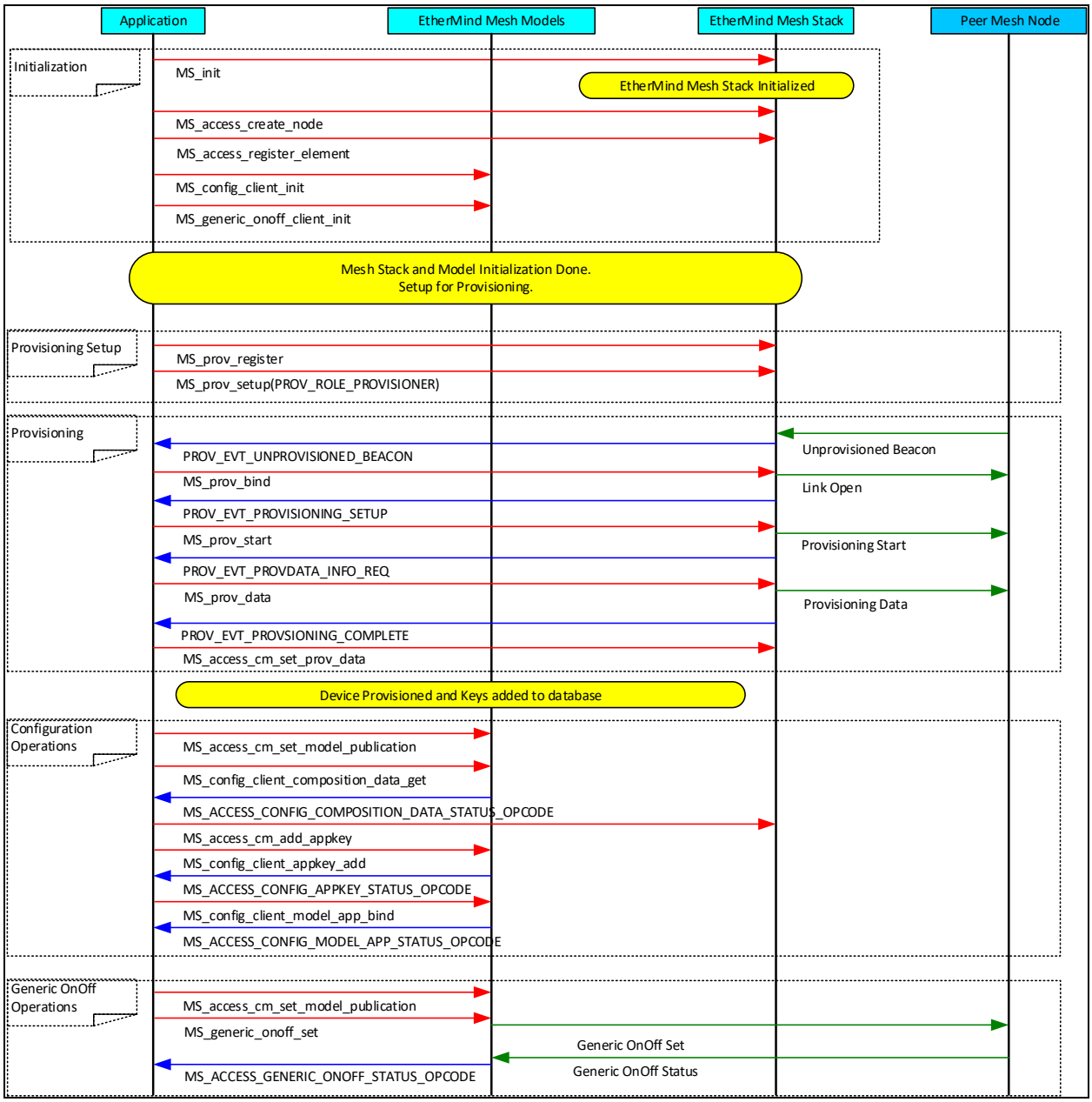
```
                        /**
                         *  NOTE:
                         *  Increment the appl_prov_data.uaddr for the next
                         *  set of device which is getting provisioned based on
                         *  the address and number of elements present in the last
                         *  provisioned device.
                         */
                    }

                    /* Set the Publish address for Config Client */
                    UI_set_publish_address(UI_prov_data.uaddr,
UI_config_client_model_handle);

                    /* Get the Composition data */
                    UI_config_client_get_composition_data(0x00);
                }
            }
            break;

        default:
            CONSOLE_OUT("Unknown Event - 0x%02X\n", event_type);
    }

    return API_SUCCESS;
}

void UI_register_prov(void)
{
    API_RESULT retval;

    CONSOLE_OUT("Registering with Provisioning layer...\n");
    retval = MS_prov_register(&UI_prov_capab, UI_prov_callback);
    CONSOLE_OUT("Retval - 0x%04X\n", retval);
}

void UI_setup_prov(UCHAR role, UCHAR brr)
{
    API_RESULT retval;

    if (PROV_ROLE_PROVISIONER == role)
    {
        CONSOLE_OUT("Setting up Provisioner for Provisioning ...\n");
        retval = MS_prov_setup
                 (
                     brr,
                     role,
                     NULL,
                     UI_PROV_SETUP_TIMEOUT_SECS
                 );

        UI_prov_role = PROV_ROLE_PROVISIONER;
    }

    CONSOLE_OUT("Retval - 0x%04X\n", retval);
}
```

Listing 7-c: Putting all together (as Model Client) Example

```
void main (void)
{
    MS_CONFIG * config_ptr;
    MS_ACCESS_NODE_ID node_id;
    MS_ACCESS_ELEMENT_DESC   element;
    MS_ACCESS_ELEMENT_HANDLE element_handle;
    API_RESULT retval;

    UCHAR role, brr;

#ifdef MS_HAVE_DYNAMIC_CONFIG
    MS_CONFIG  config;

    /* Initialize dynamic configuration */
    MS_INIT_CONFIG(config);

    config_ptr = &config;
#else
    config_ptr = NULL;
#endif /* MS_HAVE_DYNAMIC_CONFIG */

    /* Initialize OSAL */
    EM_os_init();

    /* Initialize Debug Module */
    EM_debug_init();

    /* Initialize Timer Module */
    EM_timer_init();
    timer_em_init();

    /* Initialize utilities */
    nvsto_init();

    /* Initialize Mesh Stack */
    MS_init(config_ptr);

    /* Register with underlying BLE stack */
    blebrr_register();

    /* Create Node */
    retval = MS_access_create_node(&node_id);

    /* Register Element */
    /**
     * TBD: Define GATT Namespace Descriptions from
     * https://www.bluetooth.com/specifications/assigned-numbers/gatt-namespace-
descriptors
     *
     * Using 'main' (0x0106) as Location temporarily.
     */
    element.loc = 0x0106;

    retval = MS_access_register_element
             (
                 node_id,
                 &element,
```

Listing 7-c: Putting all together (as Model Client) Example

```
                &element_handle
            );

    if (API_SUCCESS == retval)
    {
        /* Register Configuration model client */
        retval = UI_register_config_model_client(element_handle);
    }

    if (API_SUCCESS == retval)
    {
        /* Register Generic OnOff model client */
        retval = UI_register_generic_onoff_model_client(element_handle);
    }

    /* Configure as provisioner */
    UI_register_prov();

    /**
     * setup <role:[1 - Device, 2 - Provisioner]> <bearer:[1 - Adv, 2 - GATT]
     */
    role = PROV_ROLE_PROVISIONER;
    brr = PROV_BRR_ADV;
    UI_setup_prov(role, brr);

    /* Loop forever */
    MS_LOOP_FOREVER()
    {
        EM_sleep(10);
    }

    return;
}
```



How to create a Vendor Defined model?

In Bluetooth Mesh, models defines the functionality supported by the device. Bluetooth SIG specifications defines a collection of models. One can also define its own vendor-specific models to provide functionality not currently supported by standard defined models.

The below section will guide you to create your own model, using the EtherMind Mesh Stack exposed interfaces.

[1] Define Vendor Specific Opcode List

[2] Define a common opcode handler

[3] Register model with Access Layer along with opcode list and associated handler

Listing 8: Vendor defined model IDs, Opcodes and helper routines

```
#define MS_MODEL_ID_VENDOR_EXAMPLE_SERVER                          0xA001A001

#define MS_ACCESS_VENDOR_EXAMPLE_SET_OPCODE                        0xA0010001
#define MS_ACCESS_VENDOR_EXAMPLE_GET_OPCODE                        0xA0010002
#define MS_ACCESS_VENDOR_EXAMPLE_SET_UNACKNOWLEDGED_OPCODE         0xA0010003

#define MS_STATE_VENDOR_EXAMPLE_T    0xA1

/* Macro to define an empty model Opcode Handler */
#define MODEL_OPCODE_HANDLER_EMPTY_DEF(x) \
static API_RESULT (x) \
                  ( \
                      MS_ACCESS_MODEL_HANDLE * handle, \
                      MS_NET_ADDR              saddr, \
                      MS_NET_ADDR              daddr, \
                      MS_SUBNET_HANDLE         subnet_handle, \
                      MS_APPKEY_HANDLE         appkey_handle, \
                      UINT32                   opcode, \
                      UCHAR                  * data_param, \
                      UINT16                   data_len \
                  ) \
{ \
    API_RESULT retval; \
 \
    MS_IGNORE_UNUSED_PARAM(handle); \
    MS_IGNORE_UNUSED_PARAM(saddr); \
    MS_IGNORE_UNUSED_PARAM(daddr); \
    MS_IGNORE_UNUSED_PARAM(subnet_handle); \
    MS_IGNORE_UNUSED_PARAM(appkey_handle); \
    MS_IGNORE_UNUSED_PARAM(opcode); \
    MS_IGNORE_UNUSED_PARAM(data_param); \
    MS_IGNORE_UNUSED_PARAM(data_len); \
 \
    retval = API_SUCCESS; \
 \
    return retval; \
}

/* Callback Handler */
#define MODEL_OPCODE_HANDLER_CALL(handler) \
        (handler) (handle, saddr, daddr, subnet_handle, appkey_handle, opcode, data_param,
data_len)

/**
 * Vendor Example Server application Asynchronous Notification Callback.
 *
 * Vendor Example Server calls the registered callback to indicate events occurred to the
 * application.
 *
 * \param [in] ctx           Context of the message received for a specific model
instance.
 * \param [in] msg_raw       Uninterpreted/raw received message.
```

Listing 8: Vendor defined model IDs, Opcodes and helper routines

```
 * \param [in] req_type      Requested message type.
 * \param [in] state_params  Model specific state parameters.
 * \param [in] ext_params    Additional parameters.
 */
typedef API_RESULT (* MS_VENDOR_EXAMPLE_SERVER_CB)
        (
            MS_ACCESS_MODEL_REQ_MSG_CONTEXT    * ctx,
            MS_ACCESS_MODEL_REQ_MSG_RAW        * msg_raw,
            MS_ACCESS_MODEL_REQ_MSG_T          * req_type,
            MS_ACCESS_MODEL_STATE_PARAMS       * state_params,
            MS_ACCESS_MODEL_EXT_PARAMS         * ext_params

        ) DECL_REENTRANT;

typedef struct MS_state_vendor_example_struct
{
    UCHAR  value;

} MS_STATE_VENDOR_EXAMPLE_STRUCT;
```

Listing 9: Vendor defined opcode list and other global data structures

```
static DECL_CONST UINT32 vendor_example_server_opcode_list[] =
                         {
                             MS_ACCESS_VENDOR_EXAMPLE_SET_OPCODE,
                             MS_ACCESS_VENDOR_EXAMPLE_GET_OPCODE,
                             MS_ACCESS_VENDOR_EXAMPLE_SET_UNACKNOWLEDGED_OPCODE,
                         };

static MS_ACCESS_MODEL_HANDLE   vendor_example_server_model_handle;
static MS_VENDOR_EXAMPLE_SERVER_CB       vendor_example_server_UI_cb;
```

Listing 10: Vendor defined model server initialization and other helper routines Example

```
/**
 *  \brief API to send reply or to update state change
 *
 *  \par Description
 *  This is to send reply for a request or to inform change in state.
 *
 * \param [in] ctx                     Context of the message.
 * \param [in] current_state_params    Model specific current state parameters.
 * \param [in] target_state_params     Model specific target state parameters (NULL: to be
ignored).
 * \param [in] remaining_time          Time from current state to target state (0: to be
ignored).
 * \param [in] ext_params              Additional parameters (NULL: to be ignored).
 *
 *  \return API_SUCCESS or an error code indicating reason for failure
 */
API_RESULT MS_vendor_example_server_state_update
           (
               /* IN */ MS_ACCESS_MODEL_REQ_MSG_CONTEXT    * ctx,
               /* IN */ MS_ACCESS_MODEL_STATE_PARAMS       * current_state_params,
```

Listing 10: Vendor defined model server initialization and other helper routines Example

```
                /* IN */ MS_ACCESS_MODEL_STATE_PARAMS        * target_state_params,
                /* IN */ UINT16                                remaining_time,
                /* IN */ MS_ACCESS_MODEL_EXT_PARAMS          * ext_params
            )
{
    API_RESULT retval;

    /* TODO: Check what should be maximum length */
    UCHAR      buffer[32];
    UCHAR    * pdu_ptr;
    UINT16     marker;
    UINT32     opcode;

    retval = API_FAILURE;
    marker = 0;

    CONSOLE_OUT(
    "[VENDOR_EXAMPLE_SERVER] State Update.\n");

    switch (current_state_params->state_type)
    {
    }

    /* Publish - reliable */
    if (0 == marker)
    {
        pdu_ptr = NULL;
    }
    else
    {
        pdu_ptr = buffer;
    }

    retval = MS_access_reply
             (
                &ctx->handle,
                ctx->daddr,
                ctx->saddr,
                ctx->subnet_handle,
                ctx->appkey_handle,
                ACCESS_INVALID_DEFAULT_TTL,
                opcode,
                pdu_ptr,
                marker
             );

    return retval;
}

/* Empty Model Opcode Handler Defines */
MODEL_OPCODE_HANDLER_EMPTY_DEF(vendor_example_set_handler)
MODEL_OPCODE_HANDLER_EMPTY_DEF(vendor_example_get_handler)
MODEL_OPCODE_HANDLER_EMPTY_DEF(vendor_example_set_unacknowledged_handler)

/**
 * \brief Access Layer Application Asynchronous Notification Callback.
 *
```

Listing 10: Vendor defined model server initialization and other helper routines Example

```
 * \par Description
 * Access Layer calls the registered callback to indicate events occurred to the
application.
 *
 * \param [in] handle        Model Handle.
 * \param [in] saddr         16 bit Source Address.
 * \param [in] daddr         16 bit Destination Address.
 * \param [in] appkey_handle AppKey Handle.
 * \param [in] subnet_handle Subnet Handle.
 * \param [in] opcode        Opcode.
 * \param [in] data_param    Data associated with the event if any or NULL.
 * \param [in] data_len      Size of the event data. 0 if event data is NULL.
 */
API_RESULT vendor_example_server_cb
           (
                /* IN */ MS_ACCESS_MODEL_HANDLE * handle,
                /* IN */ MS_NET_ADDR              saddr,
                /* IN */ MS_NET_ADDR              daddr,
                /* IN */ MS_SUBNET_HANDLE         subnet_handle,
                /* IN */ MS_APPKEY_HANDLE         appkey_handle,
                /* IN */ UINT32                   opcode,
                /* IN */ UCHAR                  * data_param,
                /* IN */ UINT16                   data_len
           )
{
    MS_ACCESS_MODEL_REQ_MSG_CONTEXT  req_context;
    MS_ACCESS_MODEL_REQ_MSG_RAW      req_raw;
    MS_ACCESS_MODEL_REQ_MSG_T        req_type;

    MS_ACCESS_MODEL_EXT_PARAMS     * ext_params_p;

    MS_ACCESS_MODEL_STATE_PARAMS                     state_params;

    UINT16        marker;
    API_RESULT    retval;

    retval = API_SUCCESS;
    ext_params_p = NULL;

    /* Request Context */
    req_context.handle = *handle;
    req_context.saddr  = saddr;
    req_context.daddr  = daddr;
    req_context.subnet_handle = subnet_handle;
    req_context.appkey_handle = appkey_handle;

    /* Request Raw */
    req_raw.opcode = opcode;
    req_raw.data_param = data_param;
    req_raw.data_len = data_len;

    CONSOLE_OUT(
    "[VENDOR_EXAMPLE_SERVER] Callback. Opcode 0x%04X\n", opcode);

    appl_dump_bytes(data_param, data_len);

    switch(opcode)
```

Listing 10: Vendor defined model server initialization and other helper routines Example

```
    {
        case MS_ACCESS_VENDOR_EXAMPLE_GET_OPCODE:
        {
            CONSOLE_OUT(
            "MS_ACCESS_VENDOR_EXAMPLE_GET_OPCODE\n");

            MODEL_OPCODE_HANDLER_CALL(vendor_example_get_handler);

            /* Get Request Type */
            req_type.type = MS_ACCESS_MODEL_REQ_MSG_T_GET;
            req_type.to_be_acked = 0x01;

            /* Assign reqeusted state type to the application */
            state_params.state_type = MS_STATE_VENDOR_EXAMPLE_T;
        }
        break;

        case MS_ACCESS_VENDOR_EXAMPLE_SET_OPCODE:
        {
            CONSOLE_OUT(
            "MS_ACCESS_VENDOR_EXAMPLE_SET_OPCODE\n");

            MODEL_OPCODE_HANDLER_CALL(vendor_example_set_handler);

            /* Set Request Type */
            req_type.type = MS_ACCESS_MODEL_REQ_MSG_T_SET;
            req_type.to_be_acked = 0x01;

        }
        break;

        case MS_ACCESS_VENDOR_EXAMPLE_SET_UNACKNOWLEDGED_OPCODE:
        {
            CONSOLE_OUT(
            "MS_ACCESS_VENDOR_EXAMPLE_SET_UNACKNOWLEDGED_OPCODE\n");

            MODEL_OPCODE_HANDLER_CALL(vendor_example_set_unacknowledged_handler);

            /* Set Request Type */
            req_type.type = MS_ACCESS_MODEL_REQ_MSG_T_SET;
            req_type.to_be_acked = 0x00;
        }
        break;

    }

    /* Application callback */
    if (NULL != vendor_example_server_UI_cb)
    {
        vendor_example_server_UI_cb(&req_context, &req_raw, &req_type, &state_params,
ext_params_p);
    }

    return retval;
}

/**
```

Listing 10: Vendor defined model server initialization and other helper routines Example

```
 *  \brief API to initialize Vendor_Example_1 Server model
 *
 *  \par Description
 *  This is to initialize Vendor_Example_1 Server model and to register with Acess layer.
 *
 *  \param [in] element_handle
 *              Element identifier to be associated with the model instance.
 *
 *  \param [in, out] model_handle
 *                   Model identifier associated with the model instance on successful
initialization.
 *                   After power cycle of an already provisioned node, the model handle
will have
 *                   valid value and the same will be reused for registration.
 *
 *  \param [in] UI_cb    Application Callback to be used by the Vendor_Example_1 Server.
 *
 *  \return API_SUCCESS or an error code indicating reason for failure
 */
API_RESULT MS_vendor_example_server_init
           (
               /* IN */    MS_ACCESS_ELEMENT_HANDLE    element_handle,
               /* INOUT */ MS_ACCESS_MODEL_HANDLE    * model_handle,
               /* IN */    MS_VENDOR_EXAMPLE_SERVER_CB UI_cb
           )
{
    API_RESULT retval;
    MS_ACCESS_NODE_ID        node_id;
    MS_ACCESS_MODEL          model;

    /* TBD: Initialize MUTEX and other data structures */

    /* Using default node ID */
    node_id = MS_ACCESS_DEFAULT_NODE_ID;

    CONSOLE_OUT(
    "[VENDOR_EXAMPLE] Registered Element Handle 0x%02X\n", element_handle);

    /* Configure Model */
    model.model_id.id = MS_MODEL_ID_VENDOR_EXAMPLE_SERVER;
    model.model_id.type = MS_ACCESS_MODEL_TYPE_VENDOR;
    model.elem_handle = element_handle;

    /* Register Callback */
    model.cb = vendor_example_server_cb;

    /* List of Opcodes */
    model.opcodes = vendor_example_server_opcode_list;
    model.num_opcodes = sizeof(vendor_example_server_opcode_list) / sizeof(UINT32);

    retval = MS_access_register_model
             (
                 node_id,
                 &model,
                 model_handle
             );
```

Listing 10: Vendor defined model server initialization and other helper routines Example

```
    /* Save Application Callback */
    vendor_example_server_UI_cb = UI_cb;

    /* TODO: Remove */
    vendor_example_server_model_handle = *model_handle;

    return retval;
}
```

Listing 11: Vendor Defined Model state initialization and Get/Set state handlers Example

```
/** -- Vendor Defined States */
static MS_STATE_VENDOR_EXAMPLE_STRUCT UI_vendor_example;

/* Vendor Defined Model state Initialization */
void UI_vendor_defined_model_states_initialization(void)
{
    /* Vendor Defined States */
    EM_mem_set (&UI_vendor_example, 0, sizeof(UI_vendor_example));
}

/* Model state Initialization */
void UI_model_states_initialization(void)
{
    /* Generic OnOff States */
    UI_generic_onoff_model_states_initialization();

    /* Vendor Defined States */
    UI_vendor_defined_model_states_initialization();
}

/* Vendor Defined Model Get Handler */
void UI_vendor_example_model_state_get(UINT16 state_t, UINT16 state_inst, void * param,
UINT8 direction)
{
    switch(state_t)
    {
        case MS_STATE_VENDOR_EXAMPLE_T:
        {
            MS_STATE_VENDOR_EXAMPLE_STRUCT * param_p;

            param_p = (MS_STATE_VENDOR_EXAMPLE_STRUCT *)param;

            /* Ignoring Instance and direction right now */
            *param_p = UI_vendor_example;
        }
        break;

        default:
        break;
    }
}

/* Vendor Defined Model Set Handler */
void UI_vendor_example_model_state_set(UINT16 state_t, UINT16 state_inst, void * param,
```

Listing 11: Vendor Defined Model state initialization and Get/Set state handlers Example

```
UINT8 direction)
{
    switch(state_t)
    {
        case MS_STATE_VENDOR_EXAMPLE_T:
        {
            MS_STATE_VENDOR_EXAMPLE_STRUCT * param_p;

            param_p = (MS_STATE_VENDOR_EXAMPLE_STRUCT *)param;

            /* Ignoring Instance and direction right now */
            UI_vendor_example = *param_p;
        }
        break;

        default:
        break;
    }
}
```

Listing 12: Vendor Defined Model Initialization and Callback routine Example

```
/* Vendor Defined Model Server */
/**
 * \brief Server Application Asynchronous Notification Callback.
 *
 * \par Description
 * Vendor_Example_1 server calls the registered callback to indicate events occurred to
the application.
 *
 * \param [in] ctx           Context of message received for a specific model instance.
 * \param [in] msg_raw       Uninterpreted/raw received message.
 * \param [in] req_type      Requested message type.
 * \param [in] state_params  Model specific state parameters.
 * \param [in] ext_params    Additional parameters.
 */
API_RESULT UI_vendor_example_server_cb
           (
               /* IN */ MS_ACCESS_MODEL_REQ_MSG_CONTEXT    * ctx,
               /* IN */ MS_ACCESS_MODEL_REQ_MSG_RAW        * msg_raw,
               /* IN */ MS_ACCESS_MODEL_REQ_MSG_T          * req_type,
               /* IN */ MS_ACCESS_MODEL_STATE_PARAMS       * state_params,
               /* IN */ MS_ACCESS_MODEL_EXT_PARAMS         * ext_params
           )
{
    MS_STATE_VENDOR_EXAMPLE_STRUCT param;
    MS_ACCESS_MODEL_STATE_PARAMS                    current_state_params;

    API_RESULT retval;

    retval = API_SUCCESS;

    /* Check message type */
    if (MS_ACCESS_MODEL_REQ_MSG_T_GET == req_type->type)
    {
```

Listing 12: Vendor Defined Model Initialization and Callback routine Example

```
        CONSOLE_OUT(
        "[VENDOR_EXAMPLE] GET Request.\n");

        UI_vendor_example_model_state_get(state_params->state_type, 0, &param, 0);

        current_state_params.state_type = state_params->state_type;
        current_state_params.state = &param;
    }
    else if (MS_ACCESS_MODEL_REQ_MSG_T_SET == req_type->type)
    {
        CONSOLE_OUT(
        "[VENDOR_EXAMPLE] SET Request.\n");

        UI_vendor_example_model_state_set(state_params->state_type, 0,
(MS_STATE_VENDOR_EXAMPLE_STRUCT *)state_params->state, 0);

        current_state_params.state_type = state_params->state_type;
        current_state_params.state = (MS_STATE_VENDOR_EXAMPLE_STRUCT *)state_params-
>state;
    }

    /* See if to be acknowledged */
    if (0x01 == req_type->to_be_acked)
    {
        CONSOLE_OUT(
        "[VENDOR_EXAMPLE] Sending Response.\n");

        /* Parameters: Request Context, Current State, Target State (NULL: to be ignored),
Remaining Time (0: to be ignored), Additional Parameters (NULL: to be ignored) */
        retval = MS_vendor_example_server_state_update(ctx, &current_state_params, NULL,
0, NULL);
    }

    return retval;
}

API_RESULT UI_register_vendor_defined_model_server
           (
               MS_ACCESS_ELEMENT_HANDLE element_handle
           )
{
    /* Vendor Defined Server */
    MS_ACCESS_MODEL_HANDLE   UI_vendor_defined_server_model_handle;
    API_RESULT retval;

    retval = MS_vendor_example_server_init
             (
                 element_handle,
                 &UI_vendor_defined_server_model_handle,
                 UI_vendor_example_server_cb
             );

    if (API_SUCCESS == retval)
    {
        CONSOLE_OUT(
        "Vendor Defined Server Initialized. Model Handle: 0x%04X\n",
        UI_vendor_defined_server_model_handle);
```

Listing 12: Vendor Defined Model Initialization and Callback routine Example

```
    }
    else
    {
        CONSOLE_OUT(
        "[ERR] Vendor Defined Server Initialization Failed. Result: 0x%04X\n",
        retval);
    }

    return retval;
}
```

Listing 7-d: Putting all together Example

```
void main (void)
{
    MS_CONFIG * config_ptr;
    MS_ACCESS_NODE_ID node_id;
    MS_ACCESS_ELEMENT_DESC   element;
    MS_ACCESS_ELEMENT_HANDLE element_handle;
    API_RESULT retval;

    UCHAR role, brr;

#ifdef MS_HAVE_DYNAMIC_CONFIG
    MS_CONFIG  config;

    /* Initialize dynamic configuration */
    MS_INIT_CONFIG(config);

    config_ptr = &config;
#else
    config_ptr = NULL;
#endif /* MS_HAVE_DYNAMIC_CONFIG */

    /* Initialize OSAL */
    EM_os_init();

    /* Initialize Debug Module */
    EM_debug_init();

    /* Initialize Timer Module */
    EM_timer_init();
    timer_em_init();

    /* Initialize utilities */
    nvsto_init();

    /* Initialize Mesh Stack */
    MS_init(config_ptr);

    /* Register with underlying BLE stack */
    blebrr_register();

    /* Create Node */
    retval = MS_access_create_node(&node_id);
```

Listing 7-d: Putting all together Example

```
    /* Register Element */
    /**
     * TBD: Define GATT Namespace Descriptions from
     * https://www.bluetooth.com/specifications/assigned-numbers/gatt-namespace-
descriptors
     *
     * Using 'main' (0x0106) as Location temporarily.
     */
    element.loc = 0x0106;

    retval = MS_access_register_element
             (
                 node_id,
                 &element,
                 &element_handle
             );

    if (API_SUCCESS == retval)
    {
        /* Register foundation model servers */
        retval = UI_register_foundation_model_servers(element_handle);
    }

    if (API_SUCCESS == retval)
    {
        /* Register Generic OnOff model server */
        retval = UI_register_generic_onoff_model_server(element_handle);
    }

    if (API_SUCCESS == retval)
    {
        /* Register Vendor Defined model server */
        retval = UI_register_vendor_defined_model_server(element_handle);
    }

    if (API_SUCCESS == retval)
    {
        /* Initialize model states */
        UI_model_states_initialization();
    }

    /* Configure as provisionee/device */
    UI_register_prov();

    /**
     * setup <role:[1 - Device, 2 - Provisioner]> <bearer:[1 - Adv, 2 - GATT]
     */
    role = PROV_ROLE_DEVICE;
    brr = PROV_BRR_ADV;
    UI_setup_prov(role, brr);
    UI_prov_bind(brr, 0x00);

    /* Loop forever */
    MS_LOOP_FOREVER()
    {
        EM_sleep(10);
```

Listing 7-d: Putting all together Example

```
    }

    return;
}
```

How to add support for proxy feature?

Generic OnOff Server application described before, is used as the base, with following modifications

- Provisioning over GATT bearer (in place of Advertising bearer)
- Enabling proxy advertising, once the provisioning is completed
- Handling of proxy connection/disconnection events in the application

Listing 2-b: Header File Inclusion for Proxy

```
/* ---------------------------------------- Header File Inclusion */
#include "MS_common.h"
#include "MS_access_api.h"
#include "MS_config_api.h"
#include "MS_health_server_api.h"
#include "MS_generic_onoff_api.h"
#include "MS_net_api.h"
#include "blebrr.h"

/* Console Input/Output */
#define CONSOLE_OUT(...)    printf(__VA_ARGS__)
#define CONSOLE_IN(...)     scanf(__VA_ARGS__)
```

Listing 6-c: Setting up as Provisionee over GATT bearer Example

```
/* Provisionee */
/** Public Key OOB Flag */
#define UI_PROV_PUBKEY_OOBINFO                0x00

/** Static OOB Flag */
#define UI_PROV_STATIC_OOBINFO                0x00

/** Output OOB Actions Supported */
/** Currently Selecting the Output OOB Actions as Alphanumeric OOB Action */
#define UI_PROV_OUTPUT_OOB_ACTIONS PROV_MASK_OOOB_ACTION_ALPHANUMERIC

/** Output OOB Maximum size supported */
#define UI_PROV_OUTPUT_OOB_SIZE               0x04

/** Input OOB Actions supported */
#define UI_PROV_INPUT_OOB_ACTIONS             0x00

/** Input OOB Maximum size supported */
#define UI_PROV_INPUT_OOB_SIZE                0x00
```

Listing 6-c: Setting up as Provisionee over GATT bearer Example

```
/** Beacon setup timeout in seconds */
#define UI_PROV_SETUP_TIMEOUT_SECS            30

/** Attention timeout for device in seconds */
#define UI_PROV_DEVICE_ATTENTION_TIMEOUT      30

#define PROV_AUTHVAL_SIZE_PL                  16

/** Authentication values for OOB Display - To be made random */
#define UI_DISPLAY_AUTH_DIGIT                 3
#define UI_DISPLAY_AUTH_NUMERIC               35007
#define UI_DISPLAY_AUTH_STRING                "F00L"

/** Provisioning capabilities of local device */
DECL_STATIC PROV_CAPABILITIES_S UI_prov_capab =
{
    /** Number of Elements */
    0x01,

    /** Supported algorithms */
    PROV_MASK_ALGO_EC_FIPS_P256,

    /** Public key type */
    UI_PROV_PUBKEY_OOBINFO,

    /** Static OOB type */
    UI_PROV_STATIC_OOBINFO,

    /** Output OOB information */
    { UI_PROV_OUTPUT_OOB_ACTIONS, UI_PROV_OUTPUT_OOB_SIZE },

    /** Input OOB information */
    { UI_PROV_INPUT_OOB_ACTIONS, UI_PROV_INPUT_OOB_SIZE },
};

/** Unprovisioned device identifier */
DECL_STATIC PROV_DEVICE_S UI_lprov_device =
{
    /** UUID */
    {0x00, 0x11, 0x22, 0x33, 0x44, 0x55, 0x66, 0x77, 0x88, 0x99, 0xAA, 0xBB, 0xCC, 0xDD,
0xEE, 0xFF},

    /** OOB Flag */
    0x00,

    /**
     * Encoded URI Information
     * For example, to give a web address, "https://www.abc.com"
     * the URI encoded data would be -
     * 0x17 0x2F 0x2F 0x77 0x77 0x77 0x2E 0x61 0x62 0x63 0x2E 0x63 0x6F 0x6D
     * where 0x17 is the URI encoding for https:
     */
    NULL
};

/** Data exchanged during Provisiong procedure */
```

Listing 6-c: Setting up as Provisionee over GATT bearer Example

```
DECL_STATIC PROV_DATA_S UI_prov_data =
{
    /** NetKey */
    { 0x45, 0x74, 0x68, 0x65, 0x72, 0x4d, 0x69, 0x6e, 0x64, 0x4e, 0x65, 0x74, 0x4b, 0x65,
0x79, 0x00 },

    /** Index of the NetKey */
    0x0000,

    /** Flags bitmask */
    0x00,

    /** Current value of the IV index */
    0x00000001,

    /** Unicast address of the primary element */
    0x0002
};


/** Current role of application - Provisioner/Device */
DECL_STATIC UCHAR UI_prov_role;

/** Provisioning Handle */
DECL_STATIC PROV_HANDLE UI_prov_handle;

API_RESULT UI_prov_callback
           (
               PROV_HANDLE * phandle,
               UCHAR         event_type,
               API_RESULT    event_result,
               void        * event_data,
               UINT16        event_datalen
           )
{
    PROV_DATA_S * rdata;
    PROV_OOB_TYPE_S * oob_info;
    API_RESULT retval;

    UCHAR authstr[PROV_AUTHVAL_SIZE_PL << 1];
    UINT32 authnum;
    UCHAR authtype;
    UCHAR * pauth;
    UINT16 authsize;

    switch (event_type)
    {
        case PROV_EVT_PROVISIONING_SETUP:
            CONSOLE_OUT("Recvd PROV_EVT_PROVISIONING_SETUP\n");
            CONSOLE_OUT("Status - 0x%04X\n", event_result);

            /* Display the attention timeout */
            CONSOLE_OUT("Attention TImeout - %d\n", *((UCHAR *)event_data));
            break;

        case PROV_EVT_OOB_DISPLAY:
            CONSOLE_OUT("Recvd PROV_EVT_OOB_DISPLAY\n");
```

Listing 6-c: Setting up as Provisionee over GATT bearer Example

```
CONSOLE_OUT("Status - 0x%04X\n", event_result);

/* Reference the Authenticatio Type information */
oob_info = (PROV_OOB_TYPE_S *)event_data;

CONSOLE_OUT("Authenticaion Action - 0x%02X\n", oob_info->action);
CONSOLE_OUT("Authenticaion Size - 0x%02X\n", oob_info->size);

/* If role is Device, the action is of Output OOB, else Input OOB */
if (PROV_ROLE_DEVICE == UI_prov_role)
{
    if (PROV_OOOB_ACTION_ALPHANUMERIC == oob_info->action)
    {
        authtype = 1;
    }
    else if (PROV_OOOB_ACTION_NUMERIC == oob_info->action)
    {
        authtype = 2;
    }
    else
    {
        authtype = 0;
    }
}
else
{
    if (PROV_IOOB_ACTION_ALPHANUMERIC == oob_info->action)
    {
        authtype = 1;
    }
    else if (PROV_IOOB_ACTION_NUMERIC == oob_info->action)
    {
        authtype = 2;
    }
    else
    {
        authtype = 0;
    }
}

if (1 == authtype)
{
    EM_str_copy (authstr, UI_DISPLAY_AUTH_STRING);

    CONSOLE_OUT("\n\n>>> AuthVal - %s <<<\n\n", authstr);

    pauth = authstr;
    authsize = (UINT16)EM_str_len(authstr);
}
else if (2 == authtype)
{
    authnum = (UINT32)UI_DISPLAY_AUTH_NUMERIC;

    CONSOLE_OUT("\n\n>>> AuthVal - %d <<<\n\n", authnum);

    pauth = (UCHAR *)&authnum;
    authsize = sizeof(UINT32);
```

Listing 6-c: Setting up as Provisionee over GATT bearer Example

```
            }
            else
            {
                authnum = (UINT32)UI_DISPLAY_AUTH_DIGIT;

                CONSOLE_OUT("\n\n>>> AuthVal - %d <<<\n\n", authnum);

                pauth = (UCHAR *)&authnum;
                authsize = sizeof(UINT32);
            }

            /* Call to input the oob */
            CONSOLE_OUT("Setting the Authval...\n");
            retval = MS_prov_set_authval(&UI_prov_handle, pauth, authsize);
            CONSOLE_OUT("Retval - 0x%04X\n", retval);
            break;

        case PROV_EVT_OOB_ENTRY:
            CONSOLE_OUT("Recvd PROV_EVT_OOB_ENTRY\n");
            CONSOLE_OUT("Status - 0x%04X\n", event_result);

            /* Reference the Authenticatio Type information */
            oob_info = (PROV_OOB_TYPE_S *)event_data;

            CONSOLE_OUT("Authenticaion Action - 0x%02X\n", oob_info->action);
            CONSOLE_OUT("Authenticaion Size - 0x%02X\n", oob_info->size);
            break;

        case PROV_EVT_DEVINPUT_COMPLETE:
            CONSOLE_OUT("Recvd PROV_EVT_DEVINPUT_COMPLETE\n");
            CONSOLE_OUT("Status - 0x%04X\n", event_result);
            break;

        case PROV_EVT_PROVDATA_INFO:
            CONSOLE_OUT("Recvd PROV_EVT_PROVDATA_INFO\n");
            CONSOLE_OUT("Status - 0x%04X\n", event_result);

            /* Reference the Provisioning Data */
            rdata = (PROV_DATA_S *)event_data;

            CONSOLE_OUT("NetKey  : "); appl_dump_bytes(rdata->netkey,
PROV_KEY_NETKEY_SIZE);
            CONSOLE_OUT("Key ID  : 0x%04X\n", rdata->keyid);
            CONSOLE_OUT("Flags   : 0x%02X\n", rdata->flags);
            CONSOLE_OUT("IVIndex : 0x%08X\n", rdata->ivindex);
            CONSOLE_OUT("UAddr   : 0x%04X\n", rdata->uaddr);

            /* Provide Provisioning Data to Access Layer */
            MS_access_cm_set_prov_data
            (
                 rdata
            );

            break;

        case PROV_EVT_PROVISIONING_COMPLETE:
            CONSOLE_OUT("Recvd PROV_EVT_PROVISIONING_COMPLETE\n");
```

Listing 6-c: Setting up as Provisionee over GATT bearer Example

```
            CONSOLE_OUT("Status - 0x%04X\n", event_result);

            if (API_SUCCESS == event_result)
            {
                /* Already Set while handling PROV_EVT_PROVDATA_INFO */
            }
            break;

        default:
            CONSOLE_OUT("Unknown Event - 0x%02X\n", event_type);
    }

    return API_SUCCESS;
}

void UI_register_prov(void)
{
    API_RESULT retval;

    CONSOLE_OUT("Registering with Provisioning layer...\n");
    retval = MS_prov_register(&UI_prov_capab, UI_prov_callback);
    CONSOLE_OUT("Retval - 0x%04X\n", retval);
}

void UI_setup_prov(UCHAR role, UCHAR brr)
{
    API_RESULT retval;

    if (PROV_BRR_GATT == brr)
    {
        blebrr_gatt_mode_set(BLEBRR_GATT_PROV_MODE);
    }

    if (PROV_ROLE_PROVISIONER != role)
    {
        CONSOLE_OUT("Setting up Device for Provisioning ...\n");
        retval = MS_prov_setup
                 (
                     brr,
                     role,
                     &UI_lprov_device,
                     UI_PROV_SETUP_TIMEOUT_SECS
                 );

        UI_prov_role = PROV_ROLE_DEVICE;
    }
    else
    {
        CONSOLE_OUT("Setting up Provisioner for Provisioning ...\n");
        retval = MS_prov_setup
                 (
                     brr,
                     role,
                     NULL,
                     UI_PROV_SETUP_TIMEOUT_SECS
                 );
```

Listing 6-c: Setting up as Provisionee over GATT bearer Example

```
        UI_prov_role = PROV_ROLE_PROVISIONER;
    }

    CONSOLE_OUT("Retval - 0x%04X\n", retval);
}

void UI_prov_bind(UCHAR brr, UCHAR index)
{
    API_RESULT retval;

    /* Call to bind with the selected device */
    CONSOLE_OUT("Binding with the selected device...\n");
    retval = MS_prov_bind(brr, &UI_lprov_device, UI_PROV_DEVICE_ATTENTION_TIMEOUT,
&UI_prov_handle);

    CONSOLE_OUT("Retval - 0x%04X\n", retval);
}
```

Note

BLEBRR_GATT_PROXY_CHAINING feature flag shall be defined.

Listing 13: Proxy Registration, Advertising and Callback routine Example

```
void appl_proxy_start_net_id_adv(MS_SUBNET_HANDLE subnet_handle)
{
    API_RESULT retval;
    static UINT8 first_time = 0;
    if (0 == first_time)
    {
        /**
         * Register with Proxy Module as Device is going to be a Proxy.
         * This is typically a one-time-event, and hence registering the
         * PROXY when Proxy ADV is being initiated!
         */
        UI_register_proxy();

        first_time = 1;
    }

    /* Set the role to Proxy with bearer */
    blebrr_gatt_mode_set(BLEBRR_GATT_PROXY_MODE);

    CONSOLE_OUT("Start Proxy Advertisements with Network ID for Subnet 0x%04X\n",
    subnet_handle);

    retval = MS_proxy_server_adv_start
             (
                 subnet_handle,
                 MS_PROXY_NET_ID_ADV_MODE
             );

    CONSOLE_OUT("Retval - 0x%04X\n", retval);
}
```

Listing 13: Proxy Registration, Advertising and Callback routine Example

```
void UI_proxy_callback
     (
         NETIF_HANDLE       * handle,
         UCHAR                p_evt,
         UCHAR              * data_param,
         UINT16               data_len
     )
{
    UCHAR             role;

    MS_IGNORE_UNUSED_PARAM(data_len);

    switch(p_evt)
    {
        case MS_PROXY_UP_EVENT:
            CONSOLE_OUT(
            "\n\n[PROXY APPL]: MS_PROXY_UP_EVENT Received for NETIF Handle 0x%02X\n\n",
*handle);

            if (NULL != data_param)
            {
                /* Catch the current role into a local */
                role = data_param[0];

                if (BRR_SERVER_ROLE == role)
                {
                    /* Enable Proxy */
                    MS_ENABLE_PROXY_FEATURE();

                    CONSOLE_OUT(
                    "\n[PROXY APPL]: Enabling Proxy Feature!!\n");

                    /* Send Secure Network Beacons */
                    MS_net_broadcast_secure_beacon(0x0000);
                }
            }
            break;

        case MS_PROXY_DOWN_EVENT:
            CONSOLE_OUT(
            "\n\n[PROXY APPL]: MS_PROXY_DOWN_EVENT Received for NETIF Handle 0x%02X\n\n",
*handle);

            CONSOLE_OUT(
            "\n[PROXY APPL]: Disabling Proxy Feature!!\n");

            /* Disable Proxy */
            MS_DISABLE_PROXY_FEATURE();

            /**
             * Start the Proxy Advertisements on Proxy Down.
             * NOTE:
             * Need to have an cleaner bearer interface/hook which
             * informs application of the Bearer Disconnection event.
             */
            appl_proxy_start_net_id_adv(0x0000);
```

Listing 13: Proxy Registration, Advertising and Callback routine Example

```
            break;

        default:
            CONSOLE_OUT(
            "\n\n[PROXY APPL ERR]: Unknown Event Received for NETIF Handle 0x%02X!!\n\n",
*handle);
            break;
    }
}

void UI_register_proxy(void)
{
    API_RESULT retval;

    CONSOLE_OUT("Registering with Provisioning layer...\n");
    retval =  MS_proxy_register(UI_proxy_callback);
    CONSOLE_OUT("Retval - 0x%04X\n", retval);
}

API_RESULT UI_set_brr_scan_rsp_data (void)
{
    /**
     * Currently setting MT-MESH-DEMO as Complete Device Name!
     * This can be updated to each individual devices as per requirement.
     */
    UCHAR UI_brr_scanrsp_data[] =
          {
          /**
           *  Shortened Device Name: MT-MESH-SAMPLE-8
           */
           0x11, 0x09, 'M', 'T', '-', 'M', 'E', 'S', 'H', '-', 'S', 'A', 'M', 'P', 'L',
'E', '-', '8'
          };
    CONSOLE_OUT("\n Setting MT-MESH-SAMPLE-8 as Complete Device Name!\n");

    /* Set the Scan Response Data at the Bearer Layer */
    blebrr_set_adv_scanrsp_data_pl
    (
        UI_brr_scanrsp_data,
        sizeof(UI_brr_scanrsp_data)
    );

    return API_SUCCESS;
}
```

Listing 7-e: Putting all together Example

```
void main (void)
{
    MS_CONFIG * config_ptr;
    MS_ACCESS_NODE_ID node_id;
    MS_ACCESS_ELEMENT_DESC   element;
    MS_ACCESS_ELEMENT_HANDLE element_handle;
    API_RESULT retval;
```

Listing 7-e: Putting all together Example

```
    UCHAR role, brr;

#ifdef MS_HAVE_DYNAMIC_CONFIG
    MS_CONFIG  config;

    /* Initialize dynamic configuration */
    MS_INIT_CONFIG(config);

    config_ptr = &config;
#else
    config_ptr = NULL;
#endif /* MS_HAVE_DYNAMIC_CONFIG */

    /* Initialize OSAL */
    EM_os_init();

    /* Initialize Debug Module */
    EM_debug_init();

    /* Initialize Timer Module */
    EM_timer_init();
    timer_em_init();

    /* Initialize utilities */
    nvsto_init();

    /* Initialize Mesh Stack */
    MS_init(config_ptr);

    /* Register with underlying BLE stack */
    blebrr_register();

    /* Create Node */
    retval = MS_access_create_node(&node_id);

    /* Register Element */
    /**
     * TBD: Define GATT Namespace Descriptions from
     * https://www.bluetooth.com/specifications/assigned-numbers/gatt-namespace-
descriptors
     *
     * Using 'main' (0x0106) as Location temporarily.
     */
    element.loc = 0x0106;

    retval = MS_access_register_element
             (
                 node_id,
                 &element,
                 &element_handle
             );

    if (API_SUCCESS == retval)
    {
        /* Register foundation model servers */
        retval = UI_register_foundation_model_servers(element_handle);
    }
```

Listing 7-e: Putting all together Example

```
    if (API_SUCCESS == retval)
    {
        /* Register Generic OnOff model server */
        retval = UI_register_generic_onoff_model_server(element_handle);
    }

    if (API_SUCCESS == retval)
    {
        /* Initialize model states */
        UI_model_states_initialization();
    }

    /* Configure as provisionee/device */
    UI_register_prov();

    /**
     * setup <role:[1 - Device, 2 - Provisioner]> <bearer:[1 - Adv, 2 - GATT]
     */
    role = PROV_ROLE_DEVICE;
    brr  = PROV_BRR_GATT;

    /**
     * Set Scan Response Data Before Starting Provisioning.
     * This is optional/additional set of Data that the device can
     * set to enhance the User Experience.
     * For Example, set a specific device name or URL as part of the
     * Scan Response Data when awaiting connections over GATT bearer.
     */
    UI_set_brr_scan_rsp_data();

    /**
     * Setting up an Unprovisioned Device over GATT
     */
    UI_setup_prov(role, brr);

    /* Loop forever */
    MS_LOOP_FOREVER()
    {
        EM_sleep(10);
    }

    return;
}
```

How to add support for LPN feature?

Generic OnOff Server application described before, is used as the base, with following modifications

- Initiate friendship establishment once the provisioning is completed
- Handling of LPN callback events in the application

Listing 14: LPN Setup and callback routine Example

```
/* ------------------------------------ LPN parameter defines */
#define UI_FRND_CRITERIA                0x4B
#define UI_FRND_RECEIVE_DELAY_MS        100
#define UI_FRND_POLLTIMEOUT_100MS       100
#define UI_FRND_SETUPTIMEOUT            10000

void UI_lpn_seek_friend(void);
void UI_frndsetup_cb(MS_SUBNET_HANDLE subnet, UCHAR event_type, UINT16 status)
{
    API_RESULT retval;
    UINT16 num_subaddr;
    UINT16 subaddr[5];

    CONSOLE_OUT("\nFriendship Event 0x%02X on Subnet 0x%04X - 0x%04X\n",
    event_type, subnet, status);

    switch (event_type)
    {
        case MS_TRN_FRIEND_SETUP_CNF:
            CONSOLE_OUT("Recvd MS_TRN_FRIEND_SETUP_CNF - 0x%04X\n", status);

            if (API_SUCCESS == status)
            {
                /* Get the subscription list */
                num_subaddr = sizeof(subaddr) / sizeof(UINT16);
                MS_access_cm_get_all_model_subscription_list(&num_subaddr, subaddr);

                if (0 < num_subaddr)
                {
                    CONSOLE_OUT("Initiating FriendSubscriptionListAdd - %d addr\n",
                    num_subaddr);
                    retval = MS_trn_lpn_subscrn_list_add(subaddr, num_subaddr);
                    CONSOLE_OUT("Retval - 0x%04X\n", retval);
                }
            }
            else
            {
                CONSOLE_OUT("Friendship Setup Failure%04X\n", status);

                UI_lpn_seek_friend();
            }
            break;

        case MS_TRN_FRIEND_SUBSCRNLIST_CNF:
            CONSOLE_OUT("Recvd MS_TRN_FRIEND_SUBSCRNLIST_CNF - 0x%04X\n", status);
            break;

        case MS_TRN_FRIEND_CLEAR_CNF:
            CONSOLE_OUT("Recvd MS_TRN_FRIEND_CLEAR_CNF - 0x%04X\n", status);
            break;

        case MS_TRN_FRIEND_TERMINATE_IND:
                  CONSOLE_OUT("Recvd MS_TRN_FRIEND_TERMINATE_IND - 0x%04X\n", status);

            /* Enable Friend feature */
            MS_ENABLE_FRIEND_FEATURE();
```

Listing 14: LPN Setup and callback routine Example

```
                    break;

        default:
            break;
    }
}

void UI_lpn_seek_friend (void)
{
    API_RESULT retval;

    /* Enable LPN feature */
    MS_ENABLE_LPN_FEATURE();

    CONSOLE_OUT ("Requesting for friendship...\n");
    retval = MS_trn_lpn_setup_friendship
             (
                 0x00,
                 UI_FRND_CRITERIA,
                 UI_FRND_RECEIVE_DELAY_MS,
                 UI_FRND_POLLTIMEOUT_100MS,
                 UI_FRND_SETUPTIMEOUT,
                 UI_frndsetup_cb
             );
    CONSOLE_OUT ("Retval - 0x%04X\n", retval);

    return;
}
```

Listing 6-d: Seeking friendship on provisioning complete

```
...
...

API_RESULT UI_prov_callback
           (
               PROV_HANDLE * phandle,
               UCHAR         event_type,
               API_RESULT    event_result,
               void        * event_data,
               UINT16        event_datalen
           )
{
...
...
    switch (event_type)
    {
...
...
        case PROV_EVT_PROVISIONING_COMPLETE:
            CONSOLE_OUT("Recvd PROV_EVT_PROVISIONING_COMPLETE\n");
            CONSOLE_OUT("Status - 0x%04X\n", event_result);

            if (API_SUCCESS == event_result)
            {
```

Listing 6-d: Seeking friendship on provisioning complete

```
                /* Already Set while handling PROV_EVT_PROVDATA_INFO */

                /* Enable Friend feature */
                MS_ENABLE_LPN_FEATURE();

                /* Seek friendship */
                UI_lpn_seek_friend ();
            }
            break;
...
...
    }

    return API_SUCCESS;
}

...
...
```

How to add support for Friend feature?

Generic OnOff Server application described before, is used as the base, with following modifications

- Enable friend feature once the provisioning is completed

Note

- Application only need to enable Friend feature. Friend related operations are internally handled by the Mesh core stack implementation.
- Friend feature also can be enabled/disabled from the configuration client.

Listing 6-e: Enable friend feature on provisioning complete

```
...
...

API_RESULT UI_prov_callback
           (
               PROV_HANDLE * phandle,
               UCHAR         event_type,
               API_RESULT    event_result,
               void        * event_data,
               UINT16        event_datalen
           )
{
...
...
    switch (event_type)
    {
...
...
        case PROV_EVT_PROVISIONING_COMPLETE:
            CONSOLE_OUT("Recvd PROV_EVT_PROVISIONING_COMPLETE\n");
```

Listing 6-e: Enable friend feature on provisioning complete

```
            CONSOLE_OUT("Status - 0x%04X\n", event_result);

            if (API_SUCCESS == event_result)
            {
                /* Already Set while handling PROV_EVT_PROVDATA_INFO */

                /* Enable Friend feature */
                MS_ENABLE_FRIEND_FEATURE();
            }
            break;
...
...
    }

    return API_SUCCESS;
}

...
...
```

Appendix A: Flash usage and configuration for Mesh

For the operation of mesh stack following information are stored in non-volatile flash memory:

- Information exchanged during provisioning
    - Mesh Addresses
    - Encryption Keys
- Information exchanged during configuration
    - Access Model Composition
    - Access Model Configuration (AppKeys, Model Binding)
- Sequence Number used in Network Messages
- IV index and associated state for the Network

Most of these will change very rarely, except for the Sequence Number which is incremented for each new network packet transmission. If the Sequence Number is written into the flash for each increment, it can potentially make the flash non-usable in a short span of time, by reaching the write/erase cycle limit associated with a flash hardware.

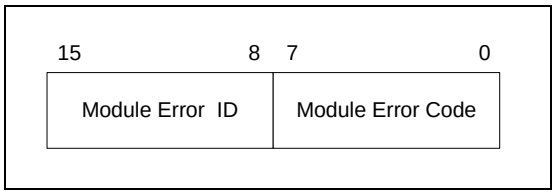
To avoid this, sequence numbers are written into the flash as a block, the size of which can be configured by the user.

Appendix B: Using EtherMind Error Codes

This section provides an informative description of EtherMind Error Codes and the design philosophies behind it.

Overview of EtherMind Error Codes

Each Error Code, from `MS_error.h`, is a 16-bit, 2-octet, `UINT16` value, as specified below:



The Module Error ID identifies the EtherMind module (Network, Transport, Bearer etc.) that is responsible for generating the error. Each module under EtherMind Mesh Core Stack & Model is given a unique Module Error ID.

The Module Error Code identifies the specific error generated by the EtherMind module (as identified by the Module Error ID field).

The definition of API_SUCCESS is 0x0000 - which is the “Success'” return value for an API returning API_RESULT data type. All other values for should be treated as errors, unless otherwise specified.

The definition of API_FAILURE is 0xFFFF - which stands for "Unknown Error Situation".

Abbreviations

Abbreviation	Reference
API	Application Programmer's Interface

References

Sl. No.	Reference
[1]	EtherMind API Document for Mesh Core Stack Modules & Models

Mindtree Limited

Global Village Campus, RVCE Post, Bangalore – 560059
INDIA
Web- www.mindtree.com