Bluetooth Mesh Generic Model Demo

This is a Application demonstrating a Bluetooth Mesh node in which
Root element has following models

- Configuration Server
- Health Server
- Generic OnOff Server
- Generic Lever Server
- Verdor Model
  > Has a single opcode for this demo

And Secondary element has following models

- Generic OnOff Server

Guides

1.Define Configure Server Information

```
static struct bt_mesh_cfg_srv cfg_srv = {
    .relay = BLE_MESH_RELAY_ENABLED,
    .beacon = BLE_MESH_BEACON_ENABLED,
#if (CONFIG_BLE_MESH_FRIEND)
    .frnd = BLE_MESH_FRIEND_ENABLED,
#else
    .frnd = BLE_MESH_FRIEND_NOT_SUPPORTED,
#endif
#if (CONFIG_BLE_MESH_PROXY)
    .gatt_proxy = BLE_MESH_GATT_PROXY_ENABLED,
#else
    .gatt_proxy = BLE_MESH_GATT_PROXY_NOT_SUPPORTED,
#endif
    .default_ttl = 7,

    /* 3 transmissions with 20ms interval */
    .net_transmit = BLE_MESH_TRANSMIT(2, 20),
    .relay_retransmit = BLE_MESH_TRANSMIT(2, 20),
};
```

- relay = `BLE_MESH_RELAY_ENABLED`

> This Demo enable local node can *Relay* message from same network.
> If you don't local node relay other message
> can use `BLE_MESH_RELAY_DISABLED` instead.

- beacon = `BLE_MESH_BEACON_ENABLED`
  > This Demo enable local node send *Secure Beacon* When provisioned.
  > If you no need local node send this information,
  > use `BLE_MESH_BEACON_DISABLED` instead.

- frnd = `BLE_MESH_FRIEND_ENABLED`
  > This Demo enable local node have a bility to become *Friend Node*.
  > If you no need local node become *Friend Node*,
  > use `BLE_MESH_FRIEND_DISABLED` instead.

- gatt_proxy = `BLE_MESH_GATT_PROXY_ENABLED`
  > This Demo enable local node have a bility beacom *Proxy Node*.
  > If you no need local node become *Proxy Node*,
  > use `BLE_MESH_GATT_PROXY_DISABLED` instead.

- default_ttl = `7`
  > This Demo use `7` as default TTL value, The maximum is 127.
  > Every message be relayed, TTL decrement by one,
  > TTL less or equal one can't be relayed.

- net_transmit = `BLE_MESH_TRANSMIT(2, 20)`
  > This Demo use *2+1=3* count transport for every Advertisering packet.
  > Caution that due to limited of version ble,
  > not permited send non-connected adv less that *100* ms.

- relay_retransmit = `BLE_MESH_TRANSMIT(2, 20)`
  > This Demo use *2+1=3* count transport for every relayed packet.
  > Caution that due to limited of version ble,
  > not permited send non-connected adv less that *100* ms.

2.Define Provision Config Information

```
static const struct bt_mesh_prov prov = {
    .uuid = dev_uuid,
    .output_size = 0,
    .output_actions = BLE_MESH_NO_OUTPUT,
    .output_number = output_number,
    .complete = prov_complete,
};
```

- uuid = `dev_uuid`
  > The UUID only used when local device unprovioned,
  > and send unprovisioned beacon information to distinct other device.
- output_size = `0`
  > This value representative out OOB capability,
  > this demo set this variable to zero, then local
  > device will select No OOB method to provisioning.
- output_actions = `BLE_MESH_NO_OUTPUT`
  > This Demo use no ouput action to provision.
- complete = `prov_complete`
  > When local device provision complete,
  > this callback function will be called.

3.Define Mesh Element Construct Information

```
static struct bt_mesh_model root_models[] = {
    BLE_MESH_MODEL_CFG_SRV(&cfg_srv),
    BLE_MESH_MODEL_HEALTH_SRV(&health_srv, &health_pub),
    BLE_MESH_MODEL_CB(BLE_MESH_MODEL_ID_GEN_ONOFF_SRV,
                      gen_onoff_op, &gen_onoff_pub, NULL, &mod_cb),
    BLE_MESH_MODEL(BLE_MESH_MODEL_ID_GEN_LEVEL_SRV,
                      gen_level_op, &gen_level_pub, NULL),
};
```

- `BLE_MESH_MODEL_CFG_SRV`
  > We use this macro definition to define Mesh model Configuration Server,
  > Cuation that, The Configuration Server only present at primary element.
- `BLE_MESH_MODEL_HEALTH_SRV`
  > We use this macro definition to define Mesh model Health Server,
  > `health_pub` as health publish construct information.
- `BLE_MESH_MODEL_CB`

> We use this macro definition to define Mesh model,
> `BLE_MESH_MODEL_ID_GEN_ONOFF_SRV` as Mesh model generic on_off server.
> `gen_onoff_op` as model opcode set, and `mod_cb`
> define model callback information.

- `BLE_MESH_MODEL`
  > This macro define have same effect with `BLE_MESH_MODEL_CB` ,
  > when no model callback provide.

4.Define Mesh Model Callback Construct

```
const static struct bt_mesh_model_cb mod_cb =
{
        .init = NULL,
#if (CONFIG_BLE_MESH_SETTINGS)
        .settings_set = settings_st,
        .settings_commit = settings_cmt,
#endif
};
```

- init = `NULL`
  > When Mesh model init, this function will be call,
  > in this demo, we unused this callback.

- settings_set = `settings_st`
  > When system power on, when store information in flash before,
  > this callback will be call. Genenrally in `settings_st` ,
  > `read_cb(cb_arg, &count, sizeof(count));` ,will be call to fetch
  > information from flash

- settings_commit = `settings_cmt`
  > When all information recovery, this callback will be called.

5.Define Mesh Verdor Model Opcodes

```
static const struct bt_mesh_model_op vnd_model_op[] = {
    {BLE_MESH_MODEL_OP_3(0x01, CID_VENDOR), 0, vnd_model_recv},
    BLE_MESH_MODEL_OP_END,
};
```

- `BLE_MESH_MODEL_OP_3`
  > Use this macro definition when define verdor model
  > opcode *0x01* as opcode, *CID_VENDOR* as company id.

> `vnd_model_recv` callback function will be call when received this message.

- `BLE_MESH_MODEL_OP_END`
  > Ending of Mesh Model opcodes defination.

6.Define Mesh Element Construct

```
static struct bt_mesh_elem elements[] = {
    BLE_MESH_ELEM(0, root_models, vnd_models),
        BLE_MESH_ELEM(0, sig1_models, BLE_MESH_MODEL_NONE),
};
```

- `BLE_MESH_ELEM`
  > Use this macro definition to define mesh element, in this example, we define two element.frist element(Primary element) has some SIG models(root_models) and verdor models. Second element only define SIG models, Caution that every element at least one model.

7. Init Bluetooth Mesh Stack

```
    err = bt_mesh_init(&prov, &comp);
    if (err)
    {
        if(err = -NOBUFS)
            BT_ERR("Memory overflow, please modify as prompted: %d",
                                                bt_mesh_ram_address_len);
        BT_ERR("Initializing mesh failed (err %d)", err);
        return;
    }
```

- `bt_mesh_init`
  > Init Bluetooth Mesh Stack, Caution that, when you edited default configuration, you must check this function returned value. All other functions should be executed after the function returns successfully.

8. Recovery From NVS Flash

```
    if (IS_ENABLED(CONFIG_BLE_MESH_SETTINGS))
    {
        settings_load();
    }
```

- `settings_load`

  > Loading Data Recovery Local Mesh Node in NVS Flash.
  > (Network Key, Application Key Element Address etc..)
  > elevant information will be obtained from FLASH.

9. Enable Provision

```
if (bt_mesh_is_provisioned())
{
    BT_INFO("Mesh network restored from flash");
}
else
{
    bt_mesh_prov_enable(BLE_MESH_PROV_ADV | BLE_MESH_PROV_GATT);
}
```

- `bt_mesh_is_provisioned`

  > This function will check current state is provisioned.

- `bt_mesh_prov_enable`

  > This function will enable provision bearer, in this example
  > we use PB-ADV `BLE_MESH_PROV_ADV` and PB-GATT `BLE_MESH_PROV_GATT` .

10. Start Publish

```
static void generic_onoff_pub_init(void)
{
    uint8_t *status;

    bt_mesh_model_msg_init(&msg, BLE_MESH_MODEL_OP_2(0x82, 0x03));
    status = net_buf_simple_add(&msg, 1);
    *status = !GPIOB_ReadPortPin(GPIO_Pin_0);
    net_buf_simple_add(&msg, 1);

    gen_onoff_pub.msg = &msg;
    gen_onoff_pub.update = update;
}
```

- `bt_mesh_model_msg_init`

  > Init Mesh model message, in this example, we use generic onoff
  > model and `gen_onoff_set_unack message` opcode.
  > In this example, we read local led(PB0) as set target value.
  > We also set `gen_onoff_pub.update = update` , which every time
  > publish, this `update` callback function will be call.

11. Store Model Data

```
static void gen_onoff_set_unack(struct bt_mesh_model *model,
                                struct bt_mesh_msg_ctx *ctx,
                                struct net_buf_simple *buf)
{
    BT_DBG("#mesh-onoff SET-UNACK");

        BT_INFO("%d",count++);

    gen_on_off_state = buf->data[0];
    gen_on_off_state == 0 ? GPIOB_SetBits(GPIO_Pin_0) : GPIOB_ResetBits(GPIO_Pin_0);
    gen_on_off_state == 0 ? GPIOB_SetBits(GPIO_Pin_22) : GPIOB_ResetBits(GPIO_Pin_22);

        if(IS_ENABLED(CONFIG_BLE_MESH_SETTINGS) && count % 10 == 0)
        {
                bt_mesh_model_data_store(model, false, &count, sizeof(count));
        }
}
```

- gen_onoff_set_unack

  > As mentioned earlier, generic onoff model periodic publish local led state,
  > when other model subscription this address(group addess, virtual address),
  > will receive this publish message. In this Demo, we just count for received
  > publish number and togger local led state.
  > Then we may be store local model state, when system power on, will recovery from flash.