RSL10 Bluetooth® Low Energy Mesh Sample Code User's Guide

M-20857-003
April 2019

© SCILLC, 2019
Previous Edition © 2018
"All Rights Reserved"

ON Semiconductor®

Table of Contents

Page

1. Introduction . . . 4
    1.1 Purpose . . . 4
    1.2 Intended Audience . . . 4
    1.3 Conventions . . . 4
    1.4 Further Reading . . . 4
2. Using the Mesh Sample Project . . . 5
    2.1 About the UART Terminal . . . 6
    2.2 Mesh Network Provisioned over PB-ADV . . . 7
        2.2.1 Goals . . . 7
        2.2.2 Setup . . . 8
            2.2.2.1 Command Line Application for Provisioner . . . 8
            2.2.2.2 Stand-alone Application for Generic OnOff Client . . . 9
            2.2.2.3 Stand-alone Application for Generic OnOff Server . . . 9
        2.2.3 Reset Provisioning Information . . . 9
        2.2.4 Provision and Configure Client Node . . . 10
        2.2.5 Provision and Configure Server Node . . . 11
        2.2.6 Test the Mesh Network . . . 12
    2.3 Mesh Network Provisioned over PB-GATT . . . 12
        2.3.1 Goals . . . 12
        2.3.2 Setup . . . 13
            2.3.2.1 Command Line Application for Provisioner . . . 13
            2.3.2.2 Stand-alone Application for Generic OnOff Client . . . 13
            2.3.2.3 Stand-alone Application for Generic OnOff Server . . . 14
        2.3.3 Provision and Configure Client Node . . . 14
        2.3.4 Provision and Configure Server Node . . . 15
        2.3.5 Test the Mesh Network . . . 16
    2.4 Mesh Network with Low Power and Friend Nodes . . . 16
        2.4.1 Goals . . . 16
        2.4.2 Stand-alone Application for Generic OnOff Server (Low Power Feature Enabled) . . . 17
        2.4.3 Provision and Configure Generic OnOff Server (Low Power Feature Enabled) . . . 17
        2.4.4 Test the Friend and Low Power Features . . . 17
        2.4.5 Adding Additional Low Power Server Nodes to the Mesh Network . . . 18
    2.5 Mesh Network with Relay nodes . . . 18
        2.5.1 Goals . . . 18
        2.5.2 Stand-alone Application for Relay Node . . . 18
        2.5.3 Stand-alone Application for Generic OnOff Client (for testing the Relay feature) . . . 19
        2.5.4 Provision and Configure the Relay Node . . . 19
        2.5.5 Provision and Configure the Generic OnOff Client (for testing the Relay feature) . . . 19
        2.5.6 Test the Relay Feature . . . 19
    2.6 Multiple Elements Node . . . 20
        2.6.1 Goals . . . 20
        2.6.2 Stand-alone Application for Multiple Element Node . . . 21

2.6.3 Provision and Configure Multiple Element Node 21
2.6.4 Test the Mesh Network 22
2.7 Mesh Network with a Multiple OnOff Client Models Node 23
2.7.1 Goals 23
2.7.2 Stand-alone Application for Multiple OnOff Client Models Node 24
2.7.3 Provision and Configure Multiple OnOff Client Models Node 25
2.7.4 Test the Mesh Network 26

CHAPTER 1
Introduction

1.1 Purpose

The *RSL10 Bluetooth® Low Energy Mesh Sample Code User's Guide* shows you what the mesh sample application (*ble_mesh*) demonstrates, how to configure the project to set up different mesh network scenarios, and how to experiment with them to verify their features and operations.

IMPORTANT: We assume you have already read *Getting Started with the RSL10 Bluetooth Low Energy Mesh* for prerequisites, installation instructions, and how to work with the sample project.

1.2 Intended Audience

This manual is for anyone interested in learning how to use RSL10 with the Bluetooth Low Energy mesh network by working with the mesh sample application.

1.3 Conventions

The following conventions are used in this manual to signify particular types of information:

`monospace`	Commands and their options, file and path names, error messages, code samples and code snippets.
`mono bold`	A placeholder for the specified information. For example, replace **`filename`** with the actual name of the file.
bold	Graphical user interface labels, such as those for menus, menu items and buttons.
italics	File names and path names, or any portion of them.

1.4 Further Reading

For more information, refer to the following documents:

- *Getting Started with the RSL10 Bluetooth Low Energy Mesh*
- *EtherMind Mesh SDK Documentation*: a compiled help file that provides the application programmer interface (API) reference material
- *EtherMind Mesh Application Developer's Guide*: provides guidelines for developing applications using the EtherMind Mesh Core Stack and Models APIs
- *EtherMind User Guide—Mesh Command Line Interface*: how to use commands to configure and control the devices on the mesh network

CHAPTER 2

Using the Mesh Sample Project

This procedure assumes you have installed the RSL10 Bluetooth Low Energy mesh support according to the instructions in *Getting Started with the RSL10 Bluetooth Low Energy Mesh*.

Table 1, below, contains a summary of all of the mesh sample applications. The name of each sample application and the features included in it are outlined below. The value of the `MESH_APP` constant (located in *ble_mesh/include/app_mesh.h*) that must be defined to build the sample application is specified, as well as the capabilities of the RSL10 board flashed with the application (ie. Friend, LPN, or Relay node). Finally, a brief description of the sample application is given.

All of the stand-alone applications, except the stand-alone provisioner, have both ADV and GATT versions. To choose the GATT application, define the `MESH_APP` constant to the version that includes `GATT` instead of `ADV`, as seen in Table 1, below. For example, `MESH_APP_SA_GATT_CLIENT` selects the GATT version of the standalone client application, while `MESH_APP_SA_ADV_CLIENT` selects the ADV version of the standalone client application. The GATT Client also has the proxy feature, which allows the Provisioner with a GATT bearer to continue to communicate with the mesh network.

Table 1. Summary of Mesh Sample Applications

Sample Application	Features	MESH_APP Define	Friend	LPN	Relay	Notes
CLI	1 Element	`MESH_APP_CLI`				By sending appropriate CLI commands to an RSL10 board programmed with this application, you can convert the board into a Provisioner or a mesh device. You can also perform any desired operations on mesh models and features.
Provisioner	1 Element Configuration Client Model	`MESH_APP_SA_ADV_PROVISIONER`				Provisions and configures a device as soon as it sees a beacon. The SW1 button on board the RSL10 is used to program publication addresses of the Client Nodes (the demonstration of this application is detailed in *Getting Started with the RSL10 Bluetooth Low Energy Mesh*.)
Client	1 Element OnOff Client Model	`MESH_APP_SA_ADV_CLIENT` `MESH_APP_SA_GATT_CLIENT`				Press the SW1 button to send an OnOff command to an OnOff Server Node.

Table 1. Summary of Mesh Sample Applications (Continued)

Sample Application	Features	MESH_APP Define	Friend	LPN	Relay	Notes
Relay Client	1 Element OnOff Client Model	MESH_APP_SA_ADV_CLIENT_RELAY MESH_APP_SA_GATT_CLIENT_RELAY			P	Press and hold the SW1 button for 2 seconds to enable the relay feature. Press the SW1 button to send an OnOff command.
Relay Tester Client	1 Element OnOff Client Model	MESH_APP_SA_ADV_CLIENT_RELAY_TESTING MESH_APP_SA_GATT_CLIENT_RELAY_TESTING				Press the SW1 button on the Client to start sending OnOff commands every 2 seconds.
Multiple OnOff Clients	2 Elements 2 OnOff Client Models	MESH_APP_SA_ADV_MULTIPLE_CLIENTS MESH_APP_SA_GATT_MULTIPLE_CLIENTS				The primary and secondary element each have one instance of the OnOff Client Model. Press the SW1 button to publish an OnOff command to an OnOff Server.
Lightness Client	1 Element Light Lightness Client Model	MESH_APP_SA_ADV_LIGHTNESS_CLIENT MESH_APP_SA_GATT_LIGHTNESS_CLIENT				Press the SW1 button to toggle the brightness of the LED on the Light Lightness Server Node.
Friend Server	1 Element OnOff Server Model	MESH_APP_SA_ADV_SERVER_FRIEND MESH_APP_SA_GATT_SERVER_FRIEND	P			Used as a basic OnOff Server (Friend feature is only used when its demonstration is required).
Low Powered Node (LPN)	1 Element OnOff Server Model	MESH_APP_SA_ADV_SERVER_LOWPOWER MESH_APP_SA_GATT_SERVER_LOWPOWER		P		Press the SW1 button to search for a Friend node. Goes into low power mode upon establishing friendship.
Relay Server	1 Element OnOff Server Model	MESH_APP_SA_ADV_SERVER_RELAY MESH_APP_SA_GATT_SERVER_RELAY			P	Press the SW1 button for 2 seconds to enable Relay feature.
Multiple Server Models	2 Elements OnOff Server Model Light Lightness Server Model	MESH_APP_SA_ADV_MULTIPLE_SERVERS MESH_APP_SA_GATT_MULTIPLE_SERVERS				The OnOff Server Model is in the primary element while the Light Lightness Server Model is in the secondary element.

2.1 About the UART Terminal

All of the mesh applications can be executed on any RSL10 Evaluation and Development Board. The UART interface is used to connect the board and the computer for issuing user commands, and for tracing activities performed by, or completed in, each mesh peer device. The onboard micro USB-B connector is used as the UART connection without additional hardware. We recommend using this connection for all applications, both stand-alone and CLI (for CLI, it is mandatory).

The computer can use any terminal emulator software to send CLI commands to, and receive responses from, the RSL10 Evaluation and Development Board. Use the Windows® device manager to search for `JLink CDC UART Port (COMXX)` and identify the COM port number your serial terminal software connects to. Also, set up the serial port at a baud rate of 115200 bps with 8 data bits, no parity, 1 stop bit, and enable the option that echoes typing (in Tera Term, this is done by selecting the "Local echo" in **Setup > Terminal**). This ensures that the terminal software displays what you enter in the terminal window; otherwise you cannot see what you are typing.

IMPORTANT: The terminal must be configured to send an end of line (EOL) sequence after each command. The EOL sequences that can be used with the CLI application are line feed (`\n`) or carriage return (`\r`). In Yet Another Terminal (YAT), the EOL sequence can be selected by going to *Terminal* > *Setting* > *Text Settings* and selecting `<LF>` or `<CR>` from the EOL sequence drop down menu.

2.2 Mesh Network Provisioned over PB-ADV

2.2.1 Goals

These instructions show how to create and set up a mesh network provisioned over PB-ADV, and operate a simple mesh network consisting of a Provisioner, a Client Node, and multiple Server Nodes, using the Generic OnOff Model. This also tests the unicast and group messaging in the mesh network. Use a CLI application for the Provisioner, as described in Section 2.2.2.1, "Command Line Application for Provisioner" on page 8, and stand-alone applications for the Client and Server Nodes.

Figure 1 shows addressing between the Client and Server Nodes, and addressing between the Client Node and a group of Server Nodes.

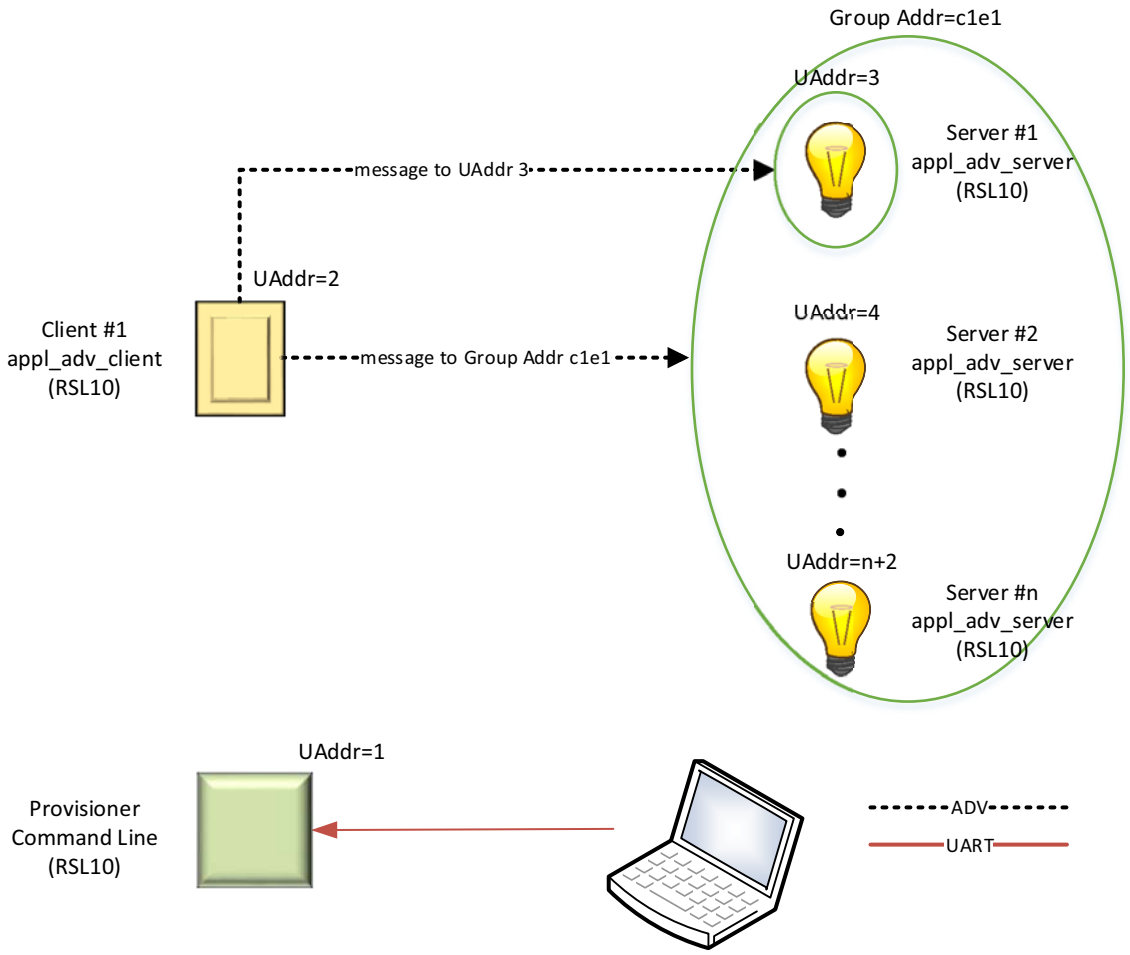


Figure 1. A Test Network with Unicast and Group Mesh Messaging

2.2.2 Setup

You need at least three RSL10 boards: one programmed with the mesh CLI application (the Provisioner), one programmed with the stand-alone Client application, and one programmed with the stand-alone Server application. You also need a computer and UART terminal software. The Provisioner must be connected to the computer using the onboard micro USB-B connector with the settings as described in Section 2.1, "About the UART Terminal" on page 6.

The next sections show the process of building, configuring and programming the boards with the required applications.

2.2.2.1 Command Line Application for Provisioner

To build a CLI application, in *ble_mesh\include\app_mesh.h* file, define the constant `MESH_APP` with the value `MESH_APP_CLI`. After building this application successfully, use it to program an RSL10 Evaluation and Development Board.

Using a USB port, connect your computer to the RSL10 Evaluation and Development Board programmed with the CLI application. Reset the Evaluation and Development Board to see the following information displayed in the UART terminal of this board:

```
**********************************
EtherMind BLE Mesh
------------------------------------------------------------
Application selected options:
------------------------------------------------------------
Command Line Interface application
------------------------------------------------------------
------------------------------------------------------------
```

IMPORTANT: The CLI arguments in the examples are described in the *EtherMind User Guide—Mesh Command Line Interface*.

2.2.2.2 Stand-alone Application for Generic OnOff Client

appl_adv_client is a stand-alone application, demonstrating a device that is provisioned over PB-ADV and uses the Generic OnOff Client Model.

To build the stand-alone Client Node application, in *ble_mesh\include\app_mesh.h* file, define the constant `MESH_APP` with the value `MESH_APP_SA_ADV_CLIENT`.

After building this application successfully, use it to program an RSL10 board, which then advertises as an Unprovisioned device.

Refer to Section 2.2.3, “Reset Provisioning Information” on page 9 to reset the provisioning information.

2.2.2.3 Stand-alone Application for Generic OnOff Server

appl_adv_server is a stand-alone application, demonstrating a device that is provisioned over PB-ADV and uses the Generic OnOff Server Model.

To build the stand-alone Server Node application, in *ble_mesh\include\app_mesh.h* file, define the constant `MESH_APP` with the value `MESH_APP_SA_ADV_SERVER_FRIEND`. The `FRIEND` suffix indicates that the Friend feature is enabled by this application. This feature is used in Section 2.4, “Mesh Network with Low Power and Friend Nodes” on page 16 to demonstrate the friendship established between a Friend Node and a Low Power Node.

After building this application successfully, use it to program an RSL10 board, which then advertises as an Unprovisioned device.

Refer to Section 2.2.3, “Reset Provisioning Information” on page 9 to reset the provisioning information.

2.2.3 Reset Provisioning Information

To prepare a device for provisioning:

1. Power on the device.
2. Press and hold the SW1 button on the device for at least five seconds, until the LED has blinked off/on five or more times. Release the button. After the provisioning information from any previous network is cleared, the LED blinks at a slow rate. Power cycle the device.

2.2.4 Provision and Configure Client Node

To prepare for provisioning and configuring the Client device:

1. Make sure that all RSL10 boards are not powered. This procedure explains when to power on and provision one device at a time.
2. Power on the RSL10 board programmed with the CLI application (built in Section 2.2.2.1, "Command Line Application for Provisioner" on page 8). For your convenience, you might want to use the USB power supply configuration for this board. Using the onboard micro USB-B port, connect this board and the computer. This board is now being used as the *Provisioner,* and further steps show how to configure it to perform this role.
3. Open an instance of the UART terminal software with the settings as described in Section 2.1, "About the UART Terminal" on page 6.
4. Power on the RSL10 board programmed with the Generic OnOff Client application (built in Section 2.2.2.2, "Stand-alone Application for Generic OnOff Client" on page 9). This board is now being used as the OnOff Client device.

In the Provisioner UART terminal, perform the following steps to provision the OnOff Client device:

1. Go to **root** > **core** > **provision**
2. Enter **setup 2 1**
3. Wait for Recvd PROV_EVT_UNPROVISIONED_BEACON reception.
4. Enter **bind 1 0**
5. Wait for **Recvd PROV_EVT_PROVISIONING_COMPLETE** with status 0x0000 (Success) in both the Provisioner and device.

The device is now provisioned and considered an OnOff Client Node. It receives its unicast address <**Addr**> as part of the Provision data sent to it from the Provisioner. In the UART terminal of the Provisioner, you can see the <**Addr**> where it is labeled `UAddr`:

```
Key ID  : 0x0000
Flags   : 0x00
IVIndex : 0x00000001
UAddr   : 0x0002
Recvd PROV_EVT_PROVISIONING_COMPLETE
```

Perform the following steps to configure the Generic OnOff Client Model of the Client Node:

1. Go to **root** > **model** > **modelc** > **config**
2. Enter **setup**
3. Enter **publishaddr** <**Addr**>
4. Enter **appkeyadd 0 0 11223344556677889900aabbccddeeff**
5. Enter **bind** <**Addr**> **0 1001 1001**

1001 is the model ID of the OnOff Client Model defined by the Bluetooth SIG. To find the rest of the model IDs, see the Bluetooth Low Energy mesh specifications.

NOTES:

- Provide all values in hexadecimal. For example, address 0x0002 is simply entered as 2.
- Because the device UUIDs in the Unprovisioned Beacon are coded with the same value, we suggest setting up RSL10 devices for provisioning and configuration one at a time.

- The bind command binds the appkey to a model with a node. The first model ID in the message (in the command above, the first 1001) is the model ID of the model in the remote node (ie. inside of the remote Client or Server Node) that the appkey is being bound to. The second model ID in the message (the second 1001) is the model ID of the local model inside of the provisioner that the appkey is being bound to. In this case, the second model ID is a dummy value because the sender (CLI provisioner) is configuring a client model, so there is no corresponding client model in the CLI provisioner being configured. If an OnOff Server Node is being configured, the command is `bind <Addr> 0 1000 1001`, which binds the appkey to the OnOff Model in the remote sever, and binds the appkey to the OnOff local Client Model in the CLI provisioner (if the OnOff Client Model has been set up in the CLI provisioner).

2.2.5 Provision and Configure Server Node

To prepare for provisioning and configuring the Server device, power on one of the RSL10 boards programmed with the Generic OnOff Server application (built in Section 2.2.2.3, “Stand-alone Application for Generic OnOff Server” on page 9). This board is now being used as the OnOff Server device.

In the Provisioner UART terminal, perform the following steps to provision the OnOff Server device:

1. Go to **root** > **core** > **provision**
2. Enter **setup 2 1**
3. Wait for **Recvd PROV_EVT_UNPROVISIONED_BEACON** reception.
4. Enter **bind 1 0**
5. Wait for **Recvd PROV_EVT_PROVISIONING_COMPLETE** with status 0x0000 (Success) in both the Provisioner and device.

The device is now Provisioned and considered an OnOff Server Node. It receives its unicast address <**Addr**> as part of the Provision data sent to it from the Provisioner. In the UART terminal of the Provisioner, you can see the <**Addr**> where it is labeled `UAddr`:

```
Key ID  : 0x0000
Flags   : 0x00
IVIndex : 0x00000001
UAddr   : 0x0003
Recvd PROV_EVT_PROVISIONING_COMPLETE
```

Continuing in the Provisioner UART terminal, perform the following steps to set up the OnOff Client Model of the CLI Provisioner:

1. Go to **root** > **model** > **modelc** > **onoff**
2. Enter **setup**

Perform the following steps to configure the OnOff Server Model of the Server Node:

1. Go to **root** > **model** > **modelc** > **config**
2. Enter **publishaddr** <**Addr**>
3. Enter **appkeyadd 0 0 112233445566778899aabbccddeeff**
4. Enter **bind** <**Addr**> **0 1000 1001**

   1000 is the model ID of the OnOff Server Model defined by the Bluetooth SIG. To find the rest of the model IDs, see the Bluetooth Low Energy mesh specification.

5. Enter **modelsubscriptionadd** <**Addr**> **c1e1 1000**

NOTE: The value `c1e1` is an example group address. Any other values satisfying the group address requirements can be used.

Repeat this procedure for each additional Generic OnOff Server device you want in the network (the setup of the OnOff Client Model in the Provisioner only has to be done once before configuring servers).

2.2.6 Test the Mesh Network

Test the mesh network by sending OnOff commands from the Client Node to the Server Nodes, using a unicast address or a group address.

Go to the Provisioner UART terminal window and use the following commands:

1. Go to **root > model > modelc** > **config**
2. Enter **publishaddr 2**
3. Enter **modelpublicationset 2 c1e1 0 0 7 0 1 0 1001**
4. On the Client Node, press the SW1 button to send a "Turn On" or "Turn Off" command. All the Servers subscribed to the 0xc1e1 group receive the command and immediately turn on/off their LEDs accordingly (this button, when pressed, alternately sends "Turn On" and "Turn Off" commands).

NOTE: The command **modelpublicationset** <**element_addr**> <**publish_addr**> 0 0 7 0 1 0 1001, sets the model publication information, where <**element_addr**> is the element address (in this example, it is the same as the unicast address of Client Node, which is 2), and <**publish_addr**> is a unicast address of a given Server Node or a group address.

5. At this point you can turn off the Provisioner and power cycle the switch (Client Node) and light (Server Node), which keep the provisioning information.

2.3 Mesh Network Provisioned over PB-GATT

2.3.1 Goals

These instructions show how to create a mesh network provisioned over PB-GATT, to set up and operate a simple mesh network consisting of a Provisioner, a switch as a Client Node, and multiple lights as Server Nodes, using the Generic OnOff Model. The procedure tests the unicast and group messaging from the Client Node to the Server Nodes. Especially, this section demonstrates the mesh Proxy Service which bridges messages between GATT and ADV bearers. A CLI application is used for the Provisioner, as described in Section 2.2.2.1, "Command Line Application for Provisioner" on page 8, and stand-alone applications for the Client and Server Nodes.

Figure 2 shows the test mesh network where the Client Node is implemented with a mesh Proxy Service which receives and retransmits mesh messages between GATT and ADV bearers.

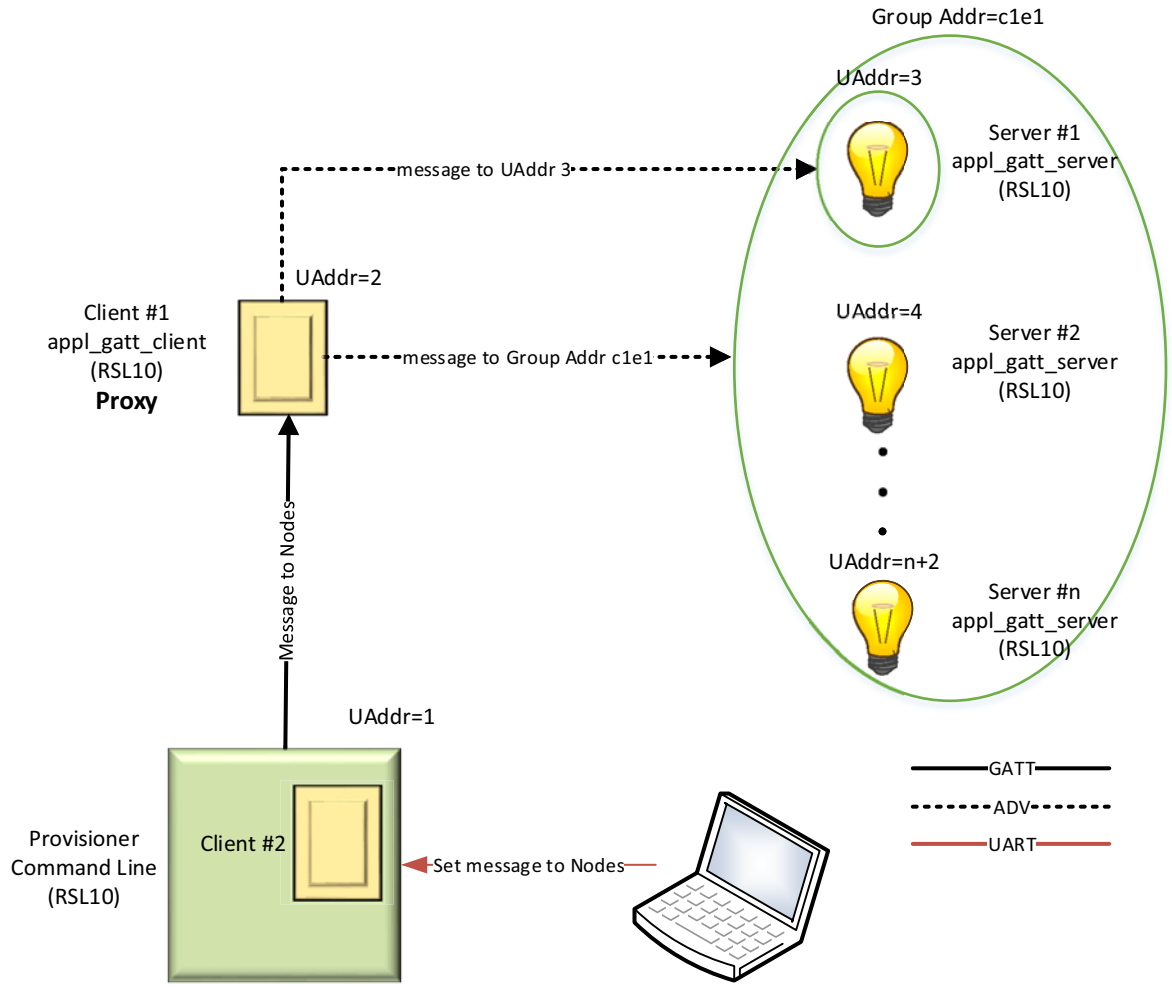


Figure 2. A Test Network with Mesh Proxy Service

2.3.2 Setup

You need at least three RSL10 boards: one programmed with the mesh CLI application (the Provisioner), one programmed with the stand-alone Client application, and one programmed with the stand-alone Server application. You also need a computer and UART terminal software. The Provisioner must be connect to the computer using the onboard micro USB-B connector with the settings as described in Section 2.1, “About the UART Terminal” on page 6.

2.3.2.1 Command Line Application for Provisioner

See Section 2.2.2.1, “Command Line Application for Provisioner” on page 8.

2.3.2.2 Stand-alone Application for Generic OnOff Client

appl_gatt_client is a stand-alone application demonstrating a device that is provisioned over PB-GATT and uses the Generic OnOff Client Model.

To build the stand-alone Client Node application, in *ble_mesh\include\app_mesh.h* file, define the constant `MESH_APP` with the value `MESH_APP_SA_GATT_CLIENT`.

After building this application successfully, use it to program an RSL10 board, which then advertises as an Unprovisioned device.

Refer to Section 2.2.3, "Reset Provisioning Information" on page 9 to reset the provisioning information.

2.3.2.3 Stand-alone Application for Generic OnOff Server

appl_gatt_server is a stand-alone application demonstrating a device that is provisioned over PB-GATT and uses the Generic OnOff Server Model.

To build the stand-alone Server Node application, in the *ble_mesh\include\app_mesh.h* file, define the constant `MESH_APP` with the value `MESH_APP_SA_GATT_SERVER_FRIEND`. The `FRIEND` suffix indicates that the Friend feature is enabled by this application. This feature is used in Section 2.4, "Mesh Network with Low Power and Friend Nodes" on page 16 to demonstrate the friendship established between a Friend Node and a Low Power Node.

After building this application successfully, use it to program an RSL10 board, which then advertises as an Unprovisioned device.

Refer to Section 2.2.3, "Reset Provisioning Information" on page 9 to reset the provisioning information.

2.3.3 Provision and Configure Client Node

To prepare for provisioning and configuring the Client device:

1. Make sure that all RSL10 boards are not powered. This procedure explains when to power on and provision one device at a time.
2. Power on the RSL10 board programmed with the CLI application (built in Section 2.2.2.1, "Command Line Application for Provisioner" on page 8). For your convenience, you might want to use the USB power supply configuration for this board. Using the onboard micro USB-B port, connect this board and the computer. This board is now being used as the *Provisioner,* and further steps show how to configure it to perform this role.
3. Open an instance of the UART terminal software with the settings as described in Section 2.1, "About the UART Terminal" on page 6.
4. Power on the RSL10 board programmed with the Generic OnOff Client application (built in Section 2.3.2.2, "Stand-alone Application for Generic OnOff Client" on page 13). This board is now being used as the OnOff Client device.

In the Provisioner UART terminal, perform the following steps to provision the OnOff Client device:

1. Go to **root > core > provision**
2. Enter **setup 2 2**

NOTE: Connectable Unprovisioned Beacons might also be received prior to setting up as Provisioner in Step 2. The Provisioner displays information regarding Unprovisioned devices that it detects to its UART terminal. Note that Bluetooth Low Energy public addresses of detected devices are displayed in hex (e.g., `60:C0:BF:03:05:D8`). However, when using those addresses in the subsequent steps, exclude the colons (:) (e.g., `60C0BF0305D8`).

3. Go to **root > brr**
4. Connect to the Client device using **connect 0** <**`BleAddr`**>, where <**`BleAddr`**> is the Bluetooth Low Energy public address of the Unprovisioned Client device. For example, **connect 0 60C0BF0305D8**

5. Discover the mesh Provisioning Service in the Unprovisioned device using **discover 0**
6. Once the mesh Provisioning Service is discovered, enable the mesh Provisioning Data Out Characteristic for Notifications using **config_ntf 1 0**
7. Wait for **Recvd PROV_EVT_PROVISIONING_COMPLETE** with status 0x0000 (Success).

NOTE: The device is now provisioned and considered an OnOff Client Node. It receives its address <**Addr**> as part of the Provision data sent to it from the Provisioner. You can see the <**Addr**> in the UART terminal of the Provisioner.

Continue with the following steps to set up mesh Proxy Service in the Client Node:

1. Disconnect the Client Node using **disconnect**
2. Connect again to the Client Node using **connect 0** <**BleAddr**>, where <**BleAddr**> is the Bluetooth Low Energy public address of Client Node.
3. Discover the mesh Proxy Service in the Provisioned device using **discover 1**
4. Once the mesh Proxy Service is discovered, enable the mesh Proxy Data Out Characteristic for Notifications using **config_ntf 1 1**
5. Wait for Proxy UP Event.

Perform the following steps to configure the OnOff Client Model of the Client Node:

1. Go to **root > model > modelc** > **config**
2. Enter **setup**
3. Enter **publishaddr** <**Addr**>, where <**Addr**> is the unicast address of Client Node.
4. Enter **appkeyadd 0 0 112233445566778899aabbccddeeff**
5. Enter **bind** <**Addr**> **0 1001 1001**, where <**Addr**> is the unicast address of Client Node.
6. Set the proxy feature of the Client Node using **proxyset 1**
7. Enter **modelpublicationset** <**element_addr**> **c1e1 0 0 7 0 1 0 1001**, where <**element_addr**> is the element address (in this example, it is the same as the unicast address of Client Node, which is 2).
8. Go to **root > brr**
9. Disconnect the Client Node using **disconnect**

2.3.4 Provision and Configure Server Node

To prepare for provisioning and configuring the Server device, power on the RSL10 board programmed with the Generic OnOff Server application built in Section 2.3.2.3, “Stand-alone Application for Generic OnOff Server” on page 14. This board is now being used as the OnOff Server device.

In the Provisioner UART terminal, perform the following steps to provision the OnOff Server device:

1. Go to **root > core > provision**
2. Enter **setup 2 2**

NOTE: Connectable Unprovisioned Beacons might also be received prior to setting up as Provisioner in Step 2.

3. Go to **root > brr**
4. Connect to the first Unprovisioned Server device using **connect 0** <**BleAddr**>, where <**BleAddr**> is the Bluetooth Low Energy public address of the first Unprovisioned Server device.
5. Discover the mesh Provisioning Service in the Unprovisioned device using **discover 0**
6. Once the mesh Provisioning Service is discovered, enable the mesh Provisioning Data Out Characteristic for Notifications using **config_ntf 1 0**

7. Wait for **Recvd PROV_EVT_PROVISIONING_COMPLETE** with status 0x0000 (Success). At this point, the first Server device is provisioned and becomes Server Node 1 of the mesh network.

Perform the following steps to set up the OnOff Model of the CLI Provisioner:

1. Go to **root** > **model** > **modelc** > **onoff**
2. Enter **setup**

Perform the following steps to configure the OnOff Server Model of the Server Node:

1. Go to **root > model > modelc > config**
2. Enter **publishaddr** <**Addr**>, where <**Addr**> is the unicast address of Server Node 1.
3. Enter **appkeyadd 0 0 112233445566778899aabbccddeeff**
4. Enter **bind** <**Addr**> **0 1000 1001**
5. Enter **modelsubscriptionadd** <**Addr**> **c1e1 1000**, where <**Addr**> is the unicast address of Server Node 1.
6. Go to **root > brr**
7. Disconnect the Server Node 1 using **disconnect**

In the Provisioner, repeat all of the above steps to provision the second Unprovisioned Server device (note that <**BleAddr**> and <**Addr**> have to be changed accordingly). At this point, the second Server device is provisioned and becomes Server Node 2 of the mesh network.

2.3.5 Test the Mesh Network

Test the mesh network by sending OnOff commands from a Generic OnOff Client Model setup in Provisioner (Client #2 in Figure 2) to the Server Nodes, using a unicast address or a group address.

Go to the Provisioner UART terminal window and do the following:

1. Go to **root > brr**
2. Connect to the mesh Proxy (co-located with the Client Node) using **connect 0** <**BleAddr**>, where <**BleAddr**> is the Bluetooth Low Energy public address of the mesh Proxy/Client Node.
3. Go to the Generic OnOff Client Model in the Provisioner by doing **root > model > modelc > onoff**
4. Enter **publishaddr** <**Addr**> **0**, where <**Addr**> is a unicast address of a Sever Node (in this example, it is 3 or 4, etc.) or a group address (in this example, it is 0xc1e1).
5. Enter **set 1 0** to send a "Turn On" command, **set 0 0** to send a "Turn Off" command. Each time the OnOff Client Model of the Provisioner sends such a command, the targeted Server Node or all Server Nodes subscribed to the targeted group receive the command and immediately turn on/off their LEDs accordingly. This demonstrates the mesh Proxy Service provided by the Client Node.

You can also press the SW1 button of the Client Node (Client #1 in Figure 2) to send "Turn On" or "Turn Off" commands to the 0xc1e1 group. All Server Nodes subscribed to this group immediately turn on/off their LEDs accordingly.

2.4 Mesh Network with Low Power and Friend Nodes

2.4.1 Goals

These instructions show how to create a mesh network with Low Power and Friend Nodes. This section assumes that you have a working mesh network with a Provisioner, a Client Node, and a Server Node with Friend feature enabled (in this case, this node is called *Friend Server Node*). The main goals in this section are to show how to add a

Server Node with the Low Power feature enabled into the network, and to demonstrate mesh operations associated with Low Power and Friend features.

NOTE: The following examples are specific to PB-ADV. To adapt the following examples to PB-GATT, change the `MESH_APP` values and steps required to provision and configure the Server Node (with Low Power feature enabled) accordingly.

2.4.2 Stand-alone Application for Generic OnOff Server (Low Power Feature Enabled)

To build the stand-alone Server Node application with Low Power feature enabled, in *ble_mesh\include\app_mesh.h* file, define the constant `MESH_APP` with the value `MESH_APP_SA_ADV_SERVER_LOWPOWER`.

After building this application successfully, use it to program an RSL10 board which then advertises as an Unprovisioned device.

Refer to Section 2.2.3, “Reset Provisioning Information” on page 9 to reset the provisioning information.

2.4.3 Provision and Configure Generic OnOff Server (Low Power Feature Enabled)

Provision and configure the Unprovisioned device in Section 2.4.2 as instructed in Section 2.2.5, “Provision and Configure Server Node” on page 11. In this case, this node is called *Low Power Server Node*.

2.4.4 Test the Friend and Low Power Features

Test these features by sending OnOff commands from the Client Node to both Server Nodes using a group address.

Go to the Provisioner UART terminal window and do the following:

1. Go to **root > model > modelc** > **config**
2. Enter **publishaddr 2**
3. Enter **modelpublicationset** 2 **c1e1 0 0 7 0 1 0 1001** (2 is assumed to be the unicast address of the Client Node).
4. On the Client Node, press the SW1 button to send a “Turn On” or “Turn Off” command. Both Server Nodes subscribed to the 0xc1e1 group receive the command and immediately turn on/off their LEDs at the same time.
5. On the Low Power Server Node, press the SW1 button to send a Friend Request command to the network. The Low Power Server and Friend Server Nodes exchange a number of messages to establish their friendship. The friendship establishment procedure takes a few seconds to complete.
6. On the Client Node, press the SW1 button to send a “Turn On” or “Turn Off” command. The Friend Server Node receives the command and immediately turns on/off its LED accordingly (it also queues this command for the Low Power Node). The Low Power Server Node, on the other hand, only receives the command after it wakes up and polls its Friend Node; therefore, it turns on/off its LED accordingly after some time delay. In the sample application, the Friend Poll Timeout is set to 10 seconds; therefore, the observed delay is typically a few seconds.

This demonstrates the friendship established between the Friend and Low Power Nodes and its associated operations. Additionally, you can use a current meter to verify that during the idle time between two consecutive Friend Poll events, the Low Power Server Node goes into sleep mode and its current consumption is much lower than the current consumption in run mode.

2.4.5 Adding Additional Low Power Server Nodes to the Mesh Network

Additional RSL10 boards configured as Low Power Server Nodes can be added to the existing mesh Network (established in the above experiment). This is done by following the same procedure presented in Section 2.4.3, "Provision and Configure Generic OnOff Server (Low Power Feature Enabled)" on page 17, for building the stand-alone Server Node application with Low Power mode enabled. Next, follow Section 2.4.3, "Provision and Configure Generic OnOff Server (Low Power Feature Enabled)" on page 17, for provisioning and configuring the Low Power Server Node.Additional friendships can then be created by following Step 5 in Section 2.4.4, "Test the Friend and Low Power Features" on page 17, to test out the additional Low Power Server Node. Note that currently, each Friend Node can only support up to five Low Power Nodes.

2.5 Mesh Network with Relay nodes

2.5.1 Goals

These instructions show how to create a mesh network with a Relay Node. This section assumes that you have a working mesh network with a CLI Provisioner and Server Node. The main goals in this section are to show how to add a Relay Node and a Client Node to test the Relay feature.

NOTE: The following examples are specific to PB-ADV. To adapt the following examples to PB-GATT, change the `MESH_APP` values and steps required to provision and configure the nodes accordingly.

Figure 3, below, illustrates the placement of the Client, Relay, and Server Nodes in order to test the Relay feature.

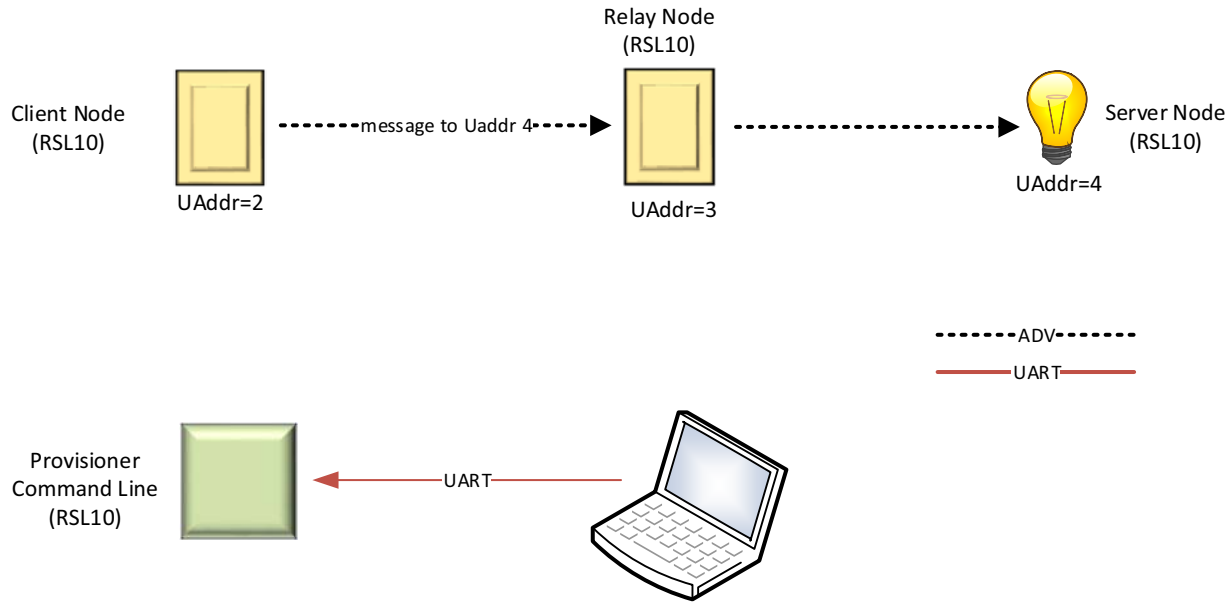


Figure 3. A Test Network with a Relay Node

2.5.2 Stand-alone Application for Relay Node

To build the stand-alone application with the Relay feature, define the constant `MESH_APP` with the value `MESH_APP_SA_ADV_CLIENT_RELAY` in the *ble_mesh\include\app_mesh.h* file.

After building this application successfully, use it to program an RSL10 board which then advertises as an Unprovisioned device.

Refer to Section 2.2.3, “Reset Provisioning Information” on page 9 to reset the provisioning information.

NOTE: You can also use a Server Node as the Relay, in which case the constant `MESH_APP` is defined with the value `MESH_APP_SA_ADV_SERVER_RELAY`. The OnOff Server and Client Models are optional for the Relay Node.

2.5.3 Stand-alone Application for Generic OnOff Client (for testing the Relay feature)

To build the stand-alone Client Node application for testing the Relay feature, define the constant `MESH_APP` with the value `MESH_APP_SA_ADV_CLIENT_RELAY_TESTING` in the *ble_mesh\include\app_mesh.h* file.

After building this application successfully, use it to program an RSL10 board which then advertises as an Unprovisioned device.

Refer to Section 2.2.3, “Reset Provisioning Information” on page 9 to reset the provisioning information.

2.5.4 Provision and Configure the Relay Node

Provision and configure the Unprovisioned device in Section 2.5.3, “Stand-alone Application for Generic OnOff Client (for testing the Relay feature)” on page 19 as instructed in Section 2.2.4, “Provision and Configure Client Node” on page 10.

2.5.5 Provision and Configure the Generic OnOff Client (for testing the Relay feature)

Provision and configure the Unprovisioned device in Section 2.5.2, “Stand-alone Application for Relay Node” on page 18 as instructed in Section 2.2.4, “Provision and Configure Client Node” on page 10.

2.5.6 Test the Relay Feature

For your convenience and to properly execute this test, you may want to use a mobile power source (eg. an external battery pack or power bank) to power the Relay Node and Server Node. First, find a location where the Server Node is out of range of the Client Node. Then, place the Relay Node in a location between the Server Node and Client Node that will allow the Relay Node to relay messages between those two nodes.

For detailed steps, go to the Provisioner UART terminal window and do the following:

1. Go to **root > model > modelc** > **config**
2. Enter **publishaddr** `<address of the Client Node>`
3. Enter **modelpublicationset** `<address of the Client Node>` `<address of Server Node>` **0 0 7 0 1 0 1001**
4. On the Client Node, press the SW1 button to begin sending OnOff commands to the Server Node every two seconds. The LED on the Server Node turns off and on every two seconds.
5. Move the Client Node and the Server Node away from each other until the LED on the Server Node stops blinking (this could be up to 100m away depending on the environment). The Server Node is now out of range of the Client Node.
6. Leave the Client and Server Nodes where they are, and put the Relay Node somewhere between them. Then, on the Relay Node, press and hold the SW1 button for 2 seconds (its LED blinks once) to enable the Relay feature. Move the Relay Node along the path between the Client and Server Nodes until the LED on the Server Node begins to blink again.

This demonstrates that the Relay feature of the Relay Node has been enabled and is working correctly.

2.6 Multiple Elements Node

2.6.1 Goals

These instructions show how to create a mesh network with a Node that contains two elements. The primary element contains a Generic OnOff Server Model, and the secondary element contains a Light Lightness Server Model. This section assumes that you have a working CLI Provisioner. The main goals in this section are to show how to add a node with multiple elements to the network, how to configure them, and how to work with the Light Lightness Model.

NOTE: The following examples are specific to PB-ADV. To adapt the following examples to PB-GATT, change the `MESH_APP` values and steps required to provision and configure the Nodes accordingly.

Figure 4, below, illustrates the addressing of the primary and secondary elements and models in the Multiple Elements Node.

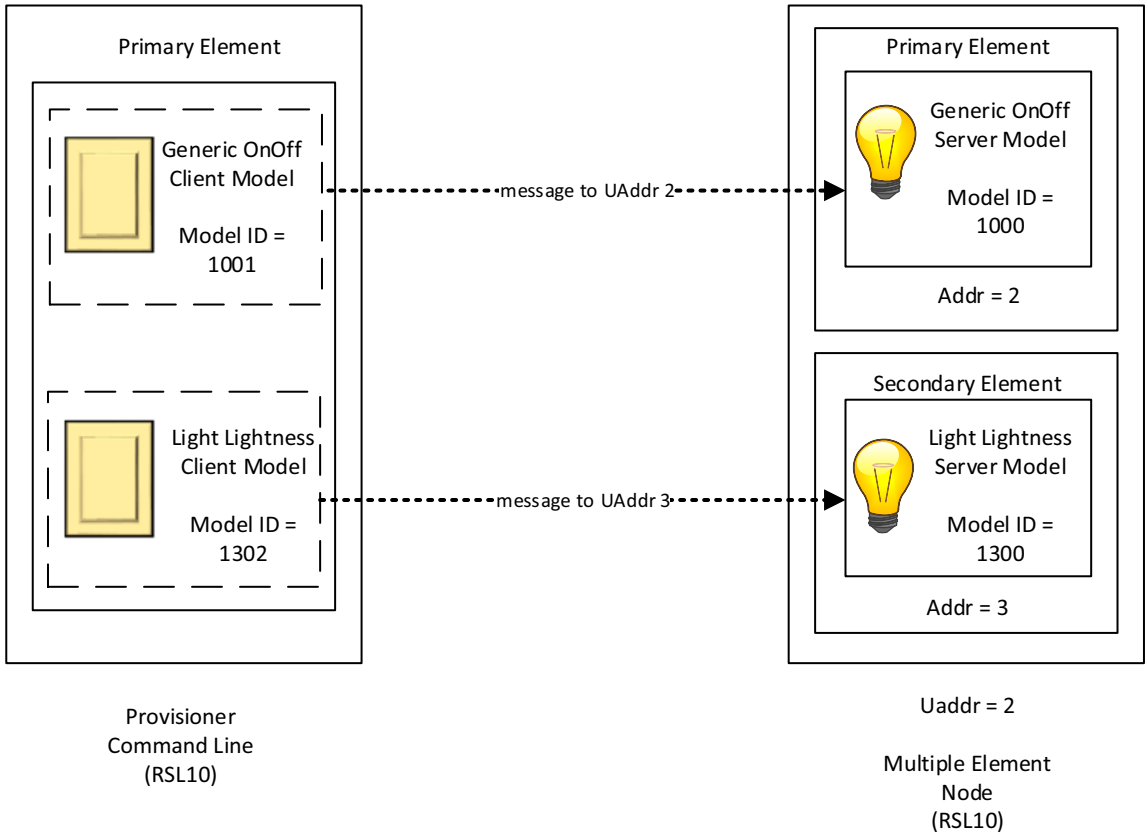


Figure 4. Mesh Network with a Multiple Elements Node

2.6.2 Stand-alone Application for Multiple Element Node

To build the stand-alone Multiple Element Node application, define the constant `MESH_APP` with the value `MESH_APP_SA_ADV_MULTIPLE_SERVERS` in the *ble_mesh\include\app_mesh.h* file.

After building this application successfully, use it to program an RSL10 board which then advertises as an Unprovisioned device.

Refer to Section 2.2.3, “Reset Provisioning Information” on page 9 to reset the provisioning information.

2.6.3 Provision and Configure Multiple Element Node

Power on the board programmed with the Multiple Element application built in Section 2.6.3, “Provision and Configure Multiple Element Node” on page 21. The board is now being used as the Multiple Element device.

In the Provisioner UART terminal, perform the following steps to provision the Multiple Element device:

1. Go to **root > core > provision**
2. Enter **Setup 2 1**
3. Wait for the **Recvd PROV_EVT_UNPROVISIONED_BEACON** reception
4. Enter **bind 1 0**
5. Wait for **Recvd PROV_EVT_PROVISIONING_COMPLETE** with status 0x0000 (Success) in the Provisioner.

The device is now provisioned. It receives its unicast address **<Addr>** as part of the provision data sent to it from the Provisioner. In the UART terminal of the Provisioner, you can see the **<Addr>** where it is labeled `UAddr`:

```
Key ID  : 0x0000
Flags   : 0x00
IVIndex : 0x00000001
UAddr   : 0x0002
Recvd PROV_EVT_PROVISIONING_COMPLETE
```

NOTE: As seen above, **<Addr>** is set to 2 and is the primary unicast of the Server Node. This address is assigned to the primary element of the node, which contains the Generic OnOff Server Model. The secondary element of the Server Node which contains the Light Lightness Server Model will be assigned to the unicast address **<Addr + 1>** (ie. 3 in this example).

Continuing in the Provisioner UART terminal, perform the following steps to set up the OnOff Client Model in the Provisioner:

1. Go to **root** > **model** > **modelc** > **onoff**
2. Enter **setup**

Perform the following steps to configure the OnOff Server Model of the Multiple Element Node:

1. Go to **root > model > modelc > config**
2. Enter **publishaddr <Addr>**
3. Enter **Appkeyadd 0 0 11223344556677889900aabbccddeeff**
4. Enter **bind <Addr> 0 1000 1001**

1000 is the model ID of the OnOff Server Model defined by the Bluetooth SIG. 1001 is the model ID of the OnOff Client Model defined by the Bluetooth SIG. To find the rest of the model IDs, see the Bluetooth Low Energy mesh specification.

NOTE: The following procedure uses the Light Lightness Client that can be set up in the CLI Provisioner. It is possible to use another device programmed with the stand-alone Light Lightness Client to send the lightness commands to the Light Lightness Server in the multiple element node. The setup would be the same as using an OnOff Client device to talk to an OnOff Server, except the model ID in commands would need to be 1302 instead of 1001. Once properly set up, the SW1 button on the Light Lightness Client toggles the brightness of the LED in three states: high, moderate, and low.

Perform the following steps to set up the Light Lightness Client Model in the Provisioner:

1. Go to **root** > **model** > **modelc** > **lightness**
2. Enter **setup**

Perform the following steps to configure the Light Lightness Server model in the Multiple Element Node:

1. Go to **root > model > modelc > config**
2. Enter **publishaddr** `<Addr>`
3. Enter **Appkeyadd 0 0 11223344556677889900aabbccddeeff**
4. Enter **bind** `<Addr + 1>` **0 1300 1302**

IMPORTANT: It is crucial that the bind command for the lightness model is addressed to the primary element's address plus one.

1300 is the model ID of the Light Lightness Server Model defined by the Bluetooth SIG. 1302 is the model ID of the Light Lightness Client Model defined by the Bluetooth SIG. To find the rest of the model IDs, see the Bluetooth Low Energy mesh specification.

2.6.4 Test the Mesh Network

Test the mesh network by sending OnOff commands from the OnOff Client model of the Provisioner, and by sending lightnessset commands from the Light Lightness Client Model of the Provisioner to the Multiple Element Node.

Go to the Provisioner UART terminal window and do the following:

1. Go to **root > model > modelc > onoff**
2. Enter **publishaddr** `<Addr>` **0**
3. Enter **set 0 0**
4. Enter **set 1 0**

Repeat the last two commands to turn the LED off and on.

Testing the Light Lightness model:

1. Go to **root > model > modelc > lightness**
2. Enter **publishaddr** `<Addr + 1>` **0**

3. Enter **lightnessset 00ff 1**
4. Enter **lightnessset ffff 1**

Repeat the last two commands to change the brightness of the LED.

2.7 Mesh Network with a Multiple OnOff Client Models Node

2.7.1 Goals

These instructions show how to create a mesh network with a Node that contains multiple instances of a given model. In this case, one Client Node contains two instances of the OnOff Client Model. This section assumes that you have a working mesh network with a CLI provisioner and two Server Nodes. The main goals in this section are to show how to add a node with two instances of the Generic OnOff Client Model to the network (in this example, the node must contain two elements, each of which containing one model instance in them), how to configure them, and how to work with them.

NOTE: The following examples are specific to PB-ADV. To adapt the following examples to PB-GATT, change the `MESH_APP` values and steps required to provision and configure the Nodes accordingly.

Figure 5, below, illustrates the addressing of two different instances of the Generic OnOff Client Model in the Multiple OnOff Client Models Node.

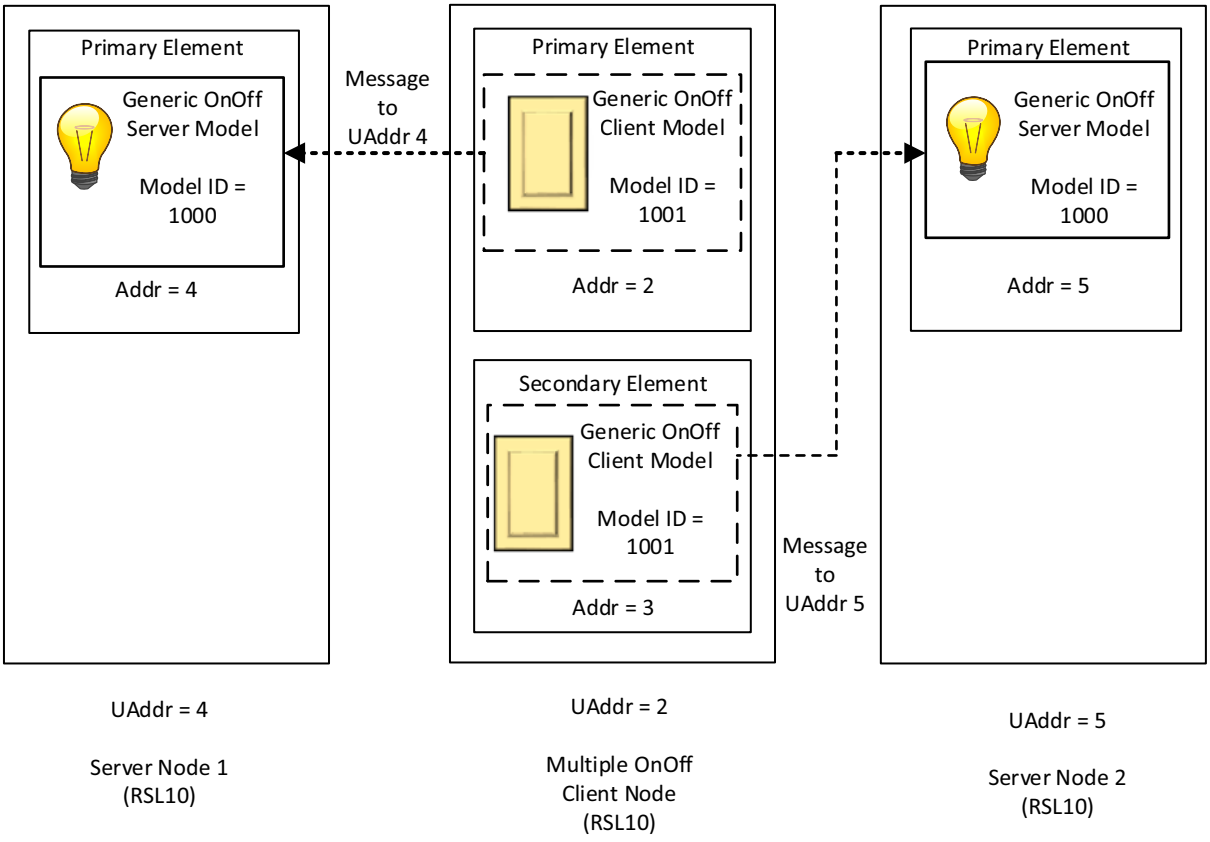


Figure 5. Mesh Network with a Multiple OnOff Client Models Node

2.7.2 Stand-alone Application for Multiple OnOff Client Models Node

To build the stand-alone Multiple OnOff Client Models Node application, define the constant `MESH_APP` with the value `MESH_APP_SA_ADV_MULTIPLE_CLIENTS` in the *ble_mesh\include\app_mesh.h* file.

After building this application successfully, use it to program an RSL10 board which then advertises as an Unprovisioned device.

Refer to Section 2.2.3, "Reset Provisioning Information" on page 9 to reset the provisioning information.

2.7.3 Provision and Configure Multiple OnOff Client Models Node

Power on the board programmed with the Multiple OnOff Client Models application built in Section 2.7.2, “Stand-alone Application for Multiple OnOff Client Models Node” on page 24. The board is now being used as the Multiple OnOff Client device.

In the Provisioner UART terminal, perform the following steps to provision the Multiple Element device:

1. Go to **root > core > provision**
2. Enter **Setup 2 1**
3. Wait for the **Recvd PROV_EVT_UNPROVISIONED_BEACON** reception
4. Enter **bind 1 0**
5. Wait for **Recvd PROV_EVT_PROVISIONING_COMPLETE** with status 0x0000 (Success) in the Provisioner.

The device is now provisioned. It receives its unicast address **<Addr>** as part of the provision data sent to it from the Provisioner. In the UART terminal of the Provisioner, you can see the **<Addr>** where it is labeled `UAddr`:

```
Key ID  : 0x0000
Flags   : 0x00
IVIndex : 0x00000001
UAddr   : 0x0002
Recvd PROV_EVT_PROVISIONING_COMPLETE
```

NOTE: As seen above, **<Addr>** is set to 2 and is the primary unicast address of the Multiple Client Node. This address is assigned to the primary element of the node, which in this case contains the first instance of the Generic OnOff Client Model. The secondary element of the Multiple Client Node which contains the second instance of the Generic OnOff Client Model will be assigned to the unicast address **<Addr + 1>** (ie. 3 in this example).

Continuing in the Provisioner UART terminal, perform the following steps to configure the first OnOff Client model:

1. Go to **root > model > modelc > config**
2. Enter **publishaddr <Addr>**
3. Enter **Appkeyadd 0 0 11223344556677889900aabbccddeeff**
4. Enter **bind <Addr> 0 1001 1001**

Perform the following steps to configure the second OnOff Client model:

1. Go to **root > model > modelc > config**
2. Enter **publishaddr <Addr>**
3. Enter **Appkeyadd 0 0 11223344556677889900aabbccddeeff**
4. Enter **bind <Addr + 1> 0 1001 1001**

1001 is the model ID of the OnOff Client Model defined by the Bluetooth SIG. To find the rest of the model IDs, see the Bluetooth Low Energy mesh specification.

IMPORTANT: It is crucial that the publishaddr is the primary element’s address, and the bind command for the second OnOff model is addressed to the primary element's address plus one.

2.7.4 Test the Mesh Network

Test the mesh network by sending OnOff commands from both of the OnOff Client Models to the Server Nodes.

Now set the publish info for the first OnOff Client Model by:

1. Go to **root > model > modelc > config**
2. Enter **publishaddr `<Addr>`**
3. Enter **modelpublicationset `<Addr>` <Addr of first Server Node> 0 0 7 0 1 0 1001**

Set the publish info for the second OnOff Client Model:

1. Go to **root > model > modelc > config**
2. Enter **publishaddr `<Addr>`**
3. Enter **modelpublicationset `<Addr + 1>` <Addr of second Server Node> 0 0 7 0 1 0 1001**

To send an OnOff command from the first Onoff Client Model, press and immediately release the SW1 button on the Multiple Client Node. The first Server Node will blinks its LED accordingly. To send an OnOff command from the second OnOff Client Model, press and hold the SW1 button for 1 second (until its LED blinks once) on the Multiple Client Node. The second Server Node will blink its LED accordingly.

BelaSigna, Ezairo, Orela, SignaKlara, and Toccata Plus are either trademarks or registered trademarks of SCILLC. All other brand names and product names appearing in this document are trademarks of their respective holders.

ON Semiconductor and  are trademarks of Semiconductor Components Industries, LLC dba ON Semiconductor or its subsidiaries in the United States and/or other countries. ON Semiconductor owns the rights to a number of patents, trademarks, copyrights, trade secrets, and other intellectual property. A listing of ON Semiconductor's product/patent coverage may be accessed at www.onsemi.com/site/pdf/Patent-Marking.pdf. ON Semiconductor reserves the right to make changes without further notice to any products herein. ON Semiconductor makes no warranty, representation or guarantee regarding the suitability of its products for any particular purpose, nor does ON Semiconductor assume any liability arising out of the application or use of any product or circuit, and specifically disclaims any and all liability, including without limitation special, consequential or incidental damages. Buyer is responsible for its products and applications using ON Semiconductor products, including compliance with all laws, regulations and safety requirements or standards, regardless of any support or applications information provided by ON Semiconductor. "Typical" parameters which may be provided in ON Semiconductor data sheets and/or specifications can and do vary in different applications and actual performance may vary over time. All operating parameters, including "Typicals" must be validated for each customer application by customer's technical experts. ON Semiconductor does not convey any license under its patent rights nor the rights of others. ON Semiconductor products are not designed, intended, or authorized for use as a critical component in life support systems or any FDA Class 3 medical devices or medical devices with a same or similar classification in a foreign jurisdiction or any devices intended for implantation in the human body. Should Buyer purchase or use ON Semiconductor products for any such unintended or unauthorized application, Buyer shall indemnify and hold ON Semiconductor and its officers, employees, subsidiaries, affiliates, and distributors harmless against all claims, costs, damages, and expenses, and reasonable attorney fees arising out of, directly or indirectly, any claim of personal injury or death associated with such unintended or unauthorized use, even if such claim alleges that ON Semiconductor was negligent regarding the design or manufacture of the part. ON Semiconductor is an Equal Opportunity/Affirmative Action Employer. This literature is subject to all applicable copyright laws and is not for resale in any manner.

PUBLICATION ORDERING INFORMATION

LITERATURE FULFILLMENT:
Literature Distribution Center for ON Semiconductor
19521 E. 32nd Pkwy, Aurora, Colorado 80011 USA
Phone: 303-675-2175 or 800-344-3860 Toll Free USA/Canada
Fax: 303-675-2176 or 800-344-3867 Toll Free USA/Canada
Email: orderlit@onsemi.com

N. American Technical Support: 800-282-9855 Toll Free USA/Canada
Europe, Middle East and Africa Technical Support: Phone: 421 33 790 2910

ON Semiconductor Website: www.onsemi.com

Order Literature: http://www.onsemi.com/orderlit

For additional information, please contact your local Sales Representative