AN1368: Bluetooth Mesh Remote Provisioning

This document describes the Mesh Remote Provisioning feature. Without it, the provisioner and the unprovisioned node must be in radio range of each other. With it, a device can be provisioned without a direct radio connection between the provisioner and the unprovisioned node. The feature uses standard Mesh networking between the provisioner and a Remote Provisioning Server, so that the server can provision any device in the radio range of the network.

KEY POINTS

- Remote Provisioning Background info
- Remote Provisioning in Bluetooth Mesh SDK
- Bluetooth Mesh Example applications
- Remote Provisioning example walkthrough

1 Background

Provisioning is a process of adding an unprovisioned device to a mesh network managed by a Provisioner. A Provisioner provides the unprovisioned device with provisioning data that allows it to become a mesh node for the initial term. The provisioning data includes a network key, the current IV Index, and the unicast address for each element.

A Provisioner is typically a smart phone or other mobile computing device, but could be also a gateway or even one of the network devices. Although only a single Provisioner is required on a network to do provisioning, multiple Provisioners may be used. The method to share cached data and coordinate across multiple Provisioners is implementation-specific.

To provision an unprovisioned device, the provisioning bearer (PB) must be established between a Provisioner and an unprovisioned device. An unprovisioned device can be identified to a Provisioner by its Device UUID and other supplementary information that may also be provided, such as some manufacturer-related information.

The PB-ADV bearer allows provisioning of an unprovisioned device by using an advertising bearer.

The PB-GATT bearer allows provisioning of an unprovisioned device by using the Mesh Provisioning Service in the GATT. For more details, refer to the Bluetooth Mesh Profile specification (1.1 draft) Chapter 5 Provisioning, the first two pages.

This document assumes that you are familiar with using Simplicity Studio to create, configure, build, and flash projects to devices. If you are not, see QSG176: Bluetooth® Mesh Quick-Start Guide for SDK v2.x and Higher. For more information about the Bluetooth mesh software components, see UG472: Bluetooth® Mesh Stack and Bluetooth® Mesh Configurator User's Guide for SDK v2.x and Higher.

2 Remote Provisioning

The PB-Remote bearer allows a Provisioner that is outside immediate radio range of an unprovisioned device to communicate with a node supporting the Remote Provisioning Server model that is within immediate radio range of the unprovisioned device, and to use that node as a re-transmitter to communicate with the unprovisioned device using PB-ADV or PB-GATT. This allows a Provisioner to provision unprovisioned devices using nodes of the mesh network.

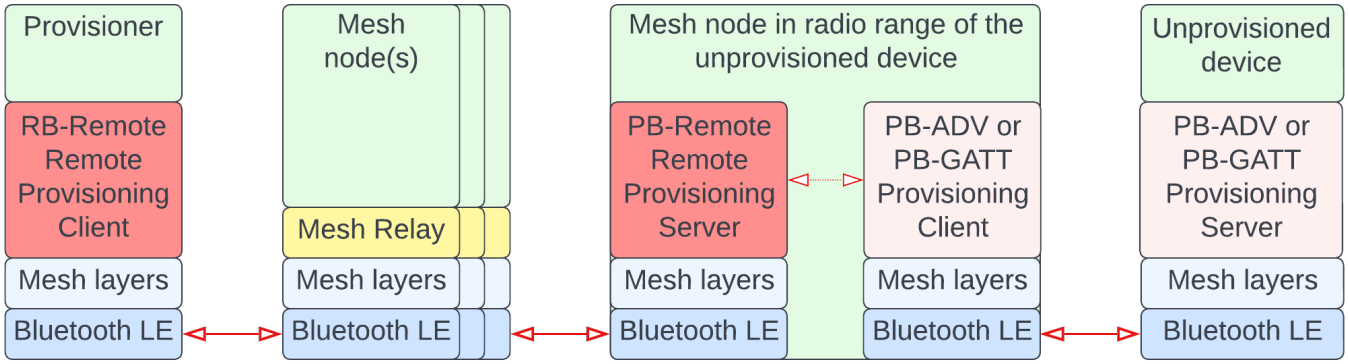


Figure 2-1. The Remote Provisioning Server Using PB-ADV or PB-GATT for Provisioning

In legacy Bluetooth mesh provisioning using PB-ADV, the Provisioner node also has the PB-ADV Provisioning Client. It can directly provision an unprovisioned device in the radio range. Bluetooth mesh provisioning using PB-GATT is similar, in that the Provisioner node also has the PB-GATT Provisioning Client (or Mesh Provisioning Service GATT Client).

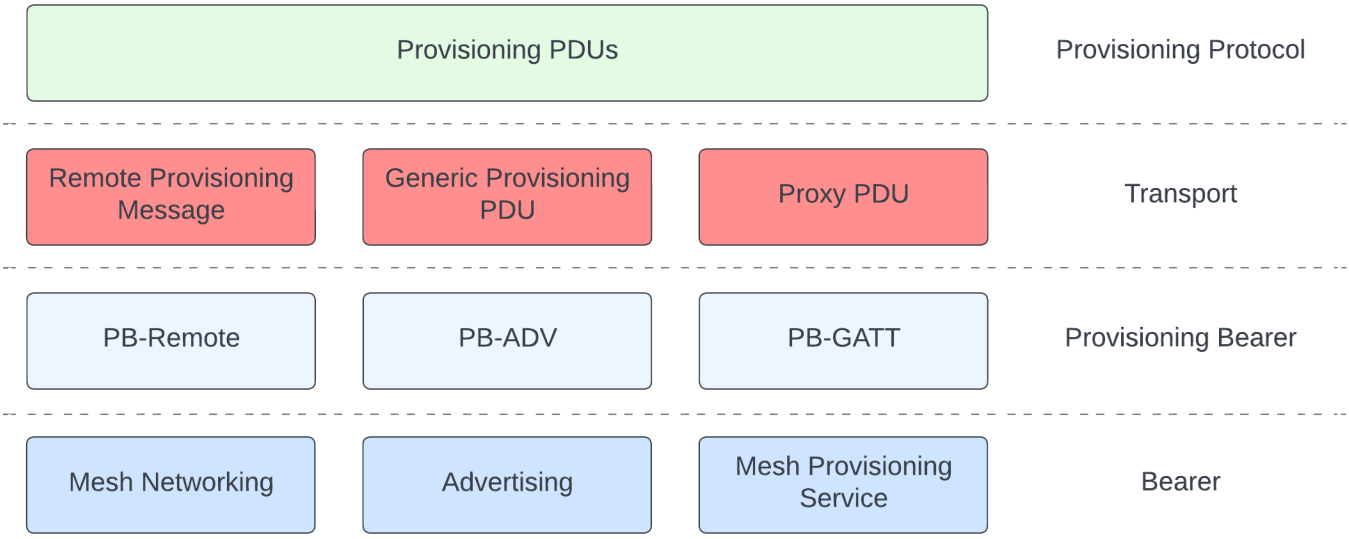


Figure 2-2. Differences Between the Provisioning Bearers

3 PB-Remote Bearer

The PB-Remote provisioning bearer uses the existing mesh network to provision an unprovisioned device that is not within immediate radio range of the Provisioner.

PB-Remote uses the PB-ADV bearer or the PB-GATT bearer for the last hop to the unprovisioned device. PB-Remote uses one of the mesh nodes as a PB-Remote Server to manage the PB-ADV or PB-GATT bearer link on behalf of the Provisioner.

Secure Provisioning should be used when PB-Remote is used. Secure Provisioning with an Authentication with No OOB or an Authentication with Static OOB of any size should be used, because these authentication methods do not require direct physical interaction with the unprovisioned device.

When PB-Remote is supported by the Provisioner, the Provisioner uses the PB-Remote Client role. When the node supports the Remote Provisioning Server model, the node uses the PB-Remote Server role.

PB-Remote may also be used to execute the Device Key Refresh procedure or the Node Address Refresh procedure or the Node Composition Refresh procedure between the Provisioner (PB-Remote Client) and the PB-Remote Server.

Multiple instances of the PB-Remote Client can be used by the Provisioner to communicate with many nodes implementing the PB-Remote Server, thus providing the capability to provision many unprovisioned devices at the same time. The PB-Remote Server can only communicate with one PB-Remote Client and can only open one supported provisioning bearer at a time.

For more details, refer to the Bluetooth Mesh Profile specification (1.1 draft) Chapter 5.2.3 PB Remote.

4 Remote Provisioning Procedures

The Provisioning procedure is used for the following purposes:

- To provision a device within immediate radio range of the Remote Provisioning Server.
- To change the Device Key Candidate of the Remote Provisioning Server by using the Device Key Refresh procedure.
- To change the Device Key Candidate and the Composition Data state of the Remote Provisioning Server by using the Node Composition Refresh procedure.
- To change the device key and the primary element address of the Remote Provisioning Server by using the Node Address Refresh procedure.

For more details, refer to the Bluetooth Mesh Profile specification (1.1 draft) Chapter 4.4.5.4 Provisioning procedure.

5 Remote Provisioning Messages

Remote Provisioning messages are used by a Remote Provisioning Client to communicate with a Remote Provisioning Server over a mesh network to find the UUID of unprovisioned devices within immediate radio range of the Remote Provisioning Server and to provision a remote unprovisioned device.

Remote Provisioning messages also can be used to obtain extended information about an unprovisioned device or to execute a Device Key Refresh procedure or a Node Address Refresh procedure or a Node Composition Refresh procedure.

The Remote Provisioning Scan Capabilities are:

1. Remote Provisioning Scan Capabilities Get
2. Remote Provisioning Scan Capabilities Status

The Remote Provisioning Scan Parameters are:

1. Remote Provisioning Scan Get
2. Remote Provisioning Scan Start
3. Remote Provisioning Scan Stop
4. Remote Provisioning Scan Status
5. Remote Provisioning Scan Report
6. Remote Provisioning Extended Scan Start
7. Remote Provisioning Extended Scan Report

The Remote Provisioning Link Parameters are

1. Remote Provisioning Link Get
2. Remote Provisioning Link Open - Can also contain `Node Provisioning Protocol Interface` (NPPI) procedures
3. Remote Provisioning Link Close
4. Remote Provisioning Link Status
5. Remote Provisioning Link Report
6. Remote Provisioning PDU Send
7. Remote Provisioning PDU Outbound Report
8. Remote Provisioning PDU Report

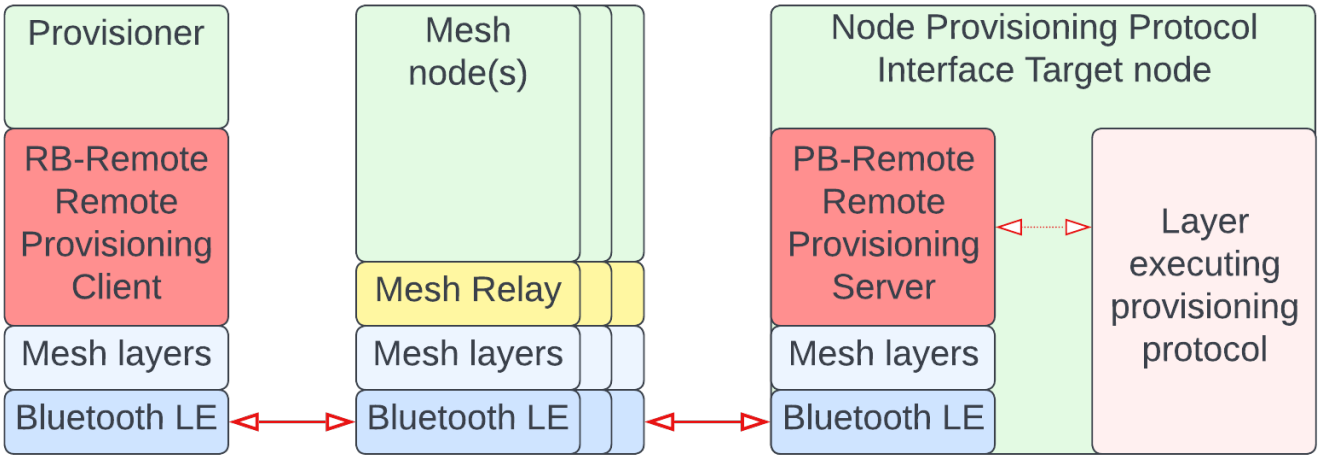
For more details, refer to the Bluetooth Mesh Profile specification (1.1 draft):

- Chapter 4.3.4 Remote Provisioning messages
- Chapter 4.2.23 Remote Provisioning Scan Capabilities
- Chapter 4.2.25 Remote Provisioning - Link Parameters

6 Node Provisioning Protocol Interface Procedures

The Node Provisioning Protocol Interface is used when a Node Provisioning Protocol Interface procedure is executed.

The Node Provisioning Protocol Interface (NPPI) is an interface used by the node to route the Provisioning PDUs between the Provisioner and the layer that is executing the provisioning protocol.



6.1 Device Key Candidate

The Device Key Candidate is a key that can replace the device key when activated. The Device Key Candidate may be delivered to the node by using an OOB mechanism or may be generated by successfully executing the Device Key Refresh procedure. When the Device Key Candidate is available, it can be activated, and replaces the device key.

6.2 Device Key Refresh Procedure

The Device Key Refresh procedure is used to change the device key (DevKey) without reprovisioning a node and without the need to reconfigure the node.

The Device Key Refresh procedure does not transfer a device key to the device over-the-air. Instead, it uses the provisioning protocol to compute the Device Key Candidate. The device key value change that results from this procedure is thus performed at the same security level as is provisioning of the unprovisioned device. The Address, NetKey, NetKey Index, and IV Index that are provided using the provisioning protocol must match the values stored on the node. The value of the Flags field is ignored.

6.3 Node Address Refresh Procedure

The Node Address Refresh procedure is used to change the node's device key and unicast address without reprovisioning.

Executing this procedure ends the current term of the node and starts a new term.

The NetKeys and AppKeys stored in the node are not removed during the procedure. Other configuration states may change (e.g., the Composition Data state may change upon successful procedure completion if features are added or removed.

6.4 Node Composition Refresh Procedure

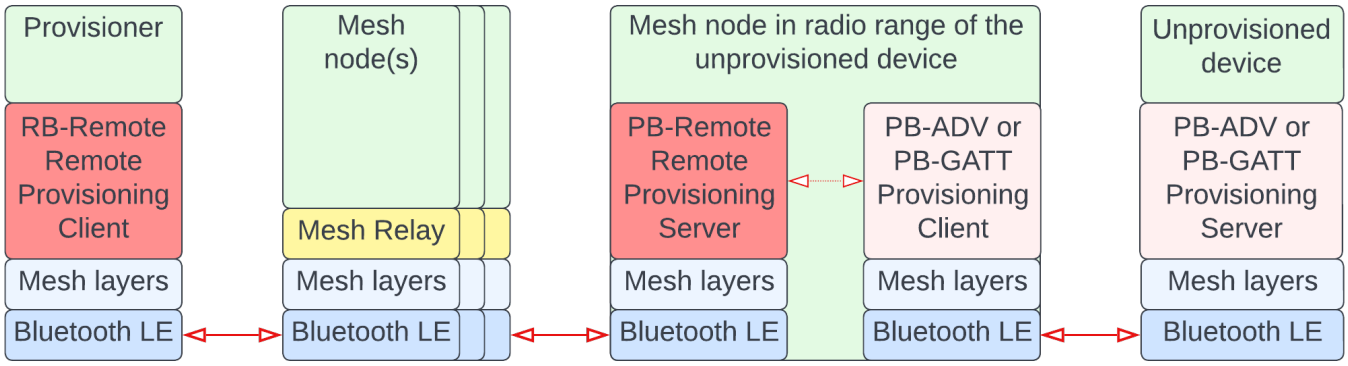
The Node Composition Refresh procedure is used to change the device key of the node and to add or delete models or features of the node without reprovisioning.

Executing this procedure ends the current term of the node and starts a new term.

Almost all states of the node remain the same during this procedure. The Composition Data state of the node is changed.

For more details, refer to the Bluetooth Mesh Profile specification (1.1 draft) Chapter 3.11.8 Node Provisioning Protocol Interface procedures.

7 Node Models and Features for Remote Provisioning



Provisioner (with Remote Provisioning support):

- Remote Provisioning Client
- Host provisioner or Embedded provisioner application

Mesh node (relaying the PB-Remote bearer messages):

- Mesh Relay just as without Remote Provisioning as the PB-Remote uses default mesh networking

Mesh node with Provisioning support (to provision the device)

- Remote Provisioning Server
- Provisioning Client, PB-ADV and/or PB-GATT
- Has to be within immediate radio range of the unprovisioned device

Unprovisioned device (to be provisioned)

- Provisioning Server, PB-ADV and/or PB-GATT

8 Bluetooth Mesh Software Components

These components can be found in the Simplicity Studio software components tab under Bluetooth Mesh -> Models -> Remote Provisioning.

Remote Provisioning Client model:

- Used to support remote provisioning client functionality of provisioning devices into a mesh network by interacting with a mesh node that supports the Remote Provisioning Server model.
- SIG Mesh Model ID: `0x0005`
- Required only in the embedded remote provisioner
- Not required in the provisioned nodes
- Only configurable feature is to Enable logging
- For more details, refer to the Bluetooth Mesh Profile specification (1.1 draft) Chapter 4.4.6 Remote Provisioning Client model.

Remote Provisioning Server model:

- Used to support remote provisioning server functionality of provisioning a remote device over the mesh network and to perform the Node Provisioning Protocol Interface procedures.
- SIG Mesh Model ID: `0x0004`
- Required in all the remote provisioning procedure-capable nodes
- Not required in the provisioned nodes
- Configurable features are Default Bearer Type and Enable logging
- For more details, refer to the Bluetooth Mesh Profile specification (1.1 draft) Chapter 4.4.5 Remote Provisioning Server model

9 Bluetooth Mesh SDK API Additions

Here is a quick look into remote provisioning additions as a list of API commands.

See *sl_btmesh_api.h* for detailed API commands and events with included documentation.

The new models are defined in *sl_btmesh_model_specification_v1_1_defs.h*. This will be integrated to *sl_btmesh_model_specification_defs.h* in the future where mesh 1.1 features are a part of a public release.

9.1 Bluetooth Mesh Stack Provisioner

`sl_btmesh_prov_provision_remote_device` - Provision device over remote provisioning server connection

9.2 Remote Provisioning Client

`sl_btmesh_remote_provisioning_client_init` - Initialize the remote provisioning client.

`sl_btmesh_remote_provisioning_client_get_scan_capabilities` - Get the value of the Remote Provisioning Scan Capabilities state.

`sl_btmesh_remote_provisioning_client_start_scan` - Start scanning for remote unprovisioned devices, within immediate radio range of the Remote Provisioning Server.

`sl_btmesh_remote_provisioning_client_get_scan_status` - Request scan status.

`sl_btmesh_remote_provisioning_client_stop_scan` - Stop Remote Provisioning Scan procedure.

`sl_btmesh_remote_provisioning_client_start_extended_scan` - Request additional information about a specific unprovisioned device or about the Remote Provisioning Server itself.

`sl_btmesh_remote_provisioning_client_open_link` – Establish the provisioning bearer between a node supporting the Remote Provisioning Server model and an unprovisioned device, or to open the Node Provisioning Protocol Interface (NPPI).

`sl_btmesh_remote_provisioning_client_get_link_status` - Get the Remote Provisioning Link state of a Remote Provisioning Server model.

`sl_btmesh_remote_provisioning_client_close_link` - Close remote provisioning link.

9.3 Remote Provisioning Server

`sl_btmesh_remote_provisioning_server_init` - Initialize the remote provisioning server.

`sl_btmesh_remote_provisioning_server_open_link` - This command is used to pass a connection handle to the remote provisioning server after having received `link_open_request` event

`sl_btmesh_remote_provisioning_server_set_default_bearer` - Set default bearer to be used in remote provisioning

10 Bluetooth Mesh Example Applications

10.1 Remote Provisioner

Bluetooth Mesh - NCP Empty 1.1

- NCP mode target application including remote provisioning support
- Compatible with parts: xG12, xG21, xG24, MGM12, xGM21, xGM24
- Only stack classes for Remote Provisioning Server and Remote Provisioning Client were added, no Models required
- Located in the GSDK folder app/bluetooth/example/btmesh_ncp_empty/btmesh_ncp_empty_v1_1.slcp

BT Mesh Host Provisioner

- PC NCP mode host application for testing host provisioning including Remote Provisioning
- to be used with **Bluetooth Mesh - NCP Empty 1.1** NCP mode target application
- Located in GSDK folder app/bluetooth/example_host/btmesh_host_provisioner
- Quick instructions: compile with a Posix-compatible toolchain with make parameters RPR=1 UI=1. See the README.txt included for more information

10.2 Remote Provisioning Relay

Any mesh application with Relay feature can relay the PB-Remote bearer messages, for example **Bluetooth Mesh - SoC Light** is a good example to test with.

10.3 Remote Provisioning Server

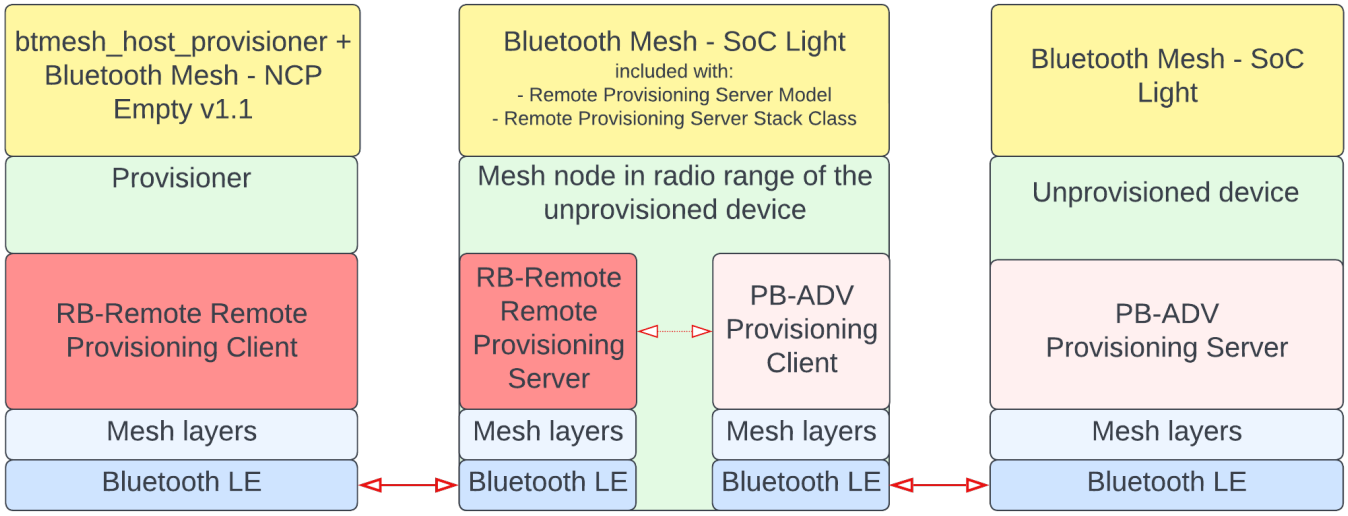
No example application is provided. Create an example application, for example **Bluetooth Mesh - SoC Light** and add the Remote Provisioning Server Model component to add mesh node Remote Provisioning Server features.

10.4 Node to be Provisioned

Any mesh node.

11 Remote Provisioning Example Walkthrough

A three- or two-device setup is used to test the Remote Provisioning feature.



11.1 Device roles

The first device is the **Provisioner** using NCP mode target application with the NCP mode host PC application or our Bluetooth Mesh smartphone application. It will provision the Unprovisioned device into the network indirectly via the Remote Provisioning Server.

The second device is the **Remote Provisioning Server** that really communicates with the Provisioner and provisions the Unprovisioned device.

The third device is the **Unprovisioned device** to be provisioned. This is a generic Bluetooth Mesh device without any additions for Remote Provisioning.

11.2 Applications

Three devices are required that can run Bluetooth Mesh (two, in the case of using a smartphone as Provisioner). Those can be any of these devices: MG12/BG12, MG21/BG21, MG24/BG24, MGM12, BGM210/MGM210, BGM240/MGM240.

Remember to flash a bootloader to the devices, either by running a demo as those include also a bootloader or by creating, building, and flashing a bootloader separately.

For more information about the bootloaders, see UG489: Silicon Labs Gecko Bootloader User's Guide for GSDK 4.0 or Higher.

11.2.1 The Provisioner, Remote Provisioning Client

If using a smartphone, omit the steps below.

The first device should run **Bluetooth Mesh - NCP Empty 1.1**. It can be the pre-built demo or a compiled example application. It does not need any changes as it already contains the Remote Provisioning Client feature.

Also compile the PC host application, **btmesh_host_provisioner** and test that it connects to the NCP target device. The readme.txt is included in the application folder app/bluetooth/example_host/btmesh_host_provisioner.

11.2.2 The Remote Provisioning Server

This device is also called the PB-Remote Server, the Provisioning Protocol executor target node, the Remote Provisioning Protocol target, the Node Provisioning Protocol Interface executor, the Provisioning Server, and so on.

The project for this device can start from a variety of examples, such as **Bluetooth Mesh - SoC Light**. Then add the component **Remote Provisioning Server Model** found under Bluetooth Mesh -> Models -> Remote Provisioning. Build and flash the application. For more information see QSG176: Bluetooth® Mesh Quick-Start Guide for SDK v2.x and Higher.

11.2.3 The Unprovisioned Device

The third device should run the **Bluetooth Mesh - SoC Light** example, or any other example. It can be the pre-built demo or a compiled example application, as it does not need any changes.

11.3 Running the Example with NCP Provisioner

When all three devices are ready, run the PC host provisioner application connected to the first Provisioner device should show a display like this:

```
Please select a functionality

1. Scan available nodes
2. Provision a beaconing node
3. List nodes in the network
4. Information about a node in the network
5. Remove node from the network
6. List available remote provisioning servers
7. Remote scan unprovisioned nodes
8. Remote provision an unprovisioned node
9. Reset provisioner node
0. Exit application
```

Select ´1´ to scan for the available unprovisioned nodes. There should be two nodes visible now. Find out before which one is the Remote Provisioning Server and which one is the Unprovisioned Device. In this example the device UUID ending bc 69 is the Remote Provisioning Server.

```
1
[I] Scanning started

[I] Unprovisioned node
[I] ID:      0
[I] UUID:    6b 11 40 f7 be 9f e5 56 95 52 30 48 20 60 bc 69
[I] OOB Capabilities: 0x0000

[I] Unprovisioned node
[I] ID:      1
[I] UUID:    6e e9 34 56 5d 73 b5 5c aa 20 31 97 10 3b 4b 84
[I] OOB Capabilities: 0x0000

[I] Scanning stopped

Press enter to continue...
```

Press ´Enter´ to continue. Then press ´2´ to provision the Remote Provisioning Server node. This example selects the unprovisioned node with ID ´0´. The following is displayed:

```
0
[I] Provisioning...
[I] Device provisioned
[I] UUID:    6b 11 40 f7 be 9f e5 56 95 52 30 48 20 60 bc 69
[I] Address: 0x20005

Configuration of node (netkey_idx=0,addr=0x2005) is started.
DCD query of node (netkey_idx=0,addr=0x2005) completed.
Node (netkey_idx=0,addr=0x2005) runs btmesh_soc_light example.
[E] Status: model_attr_status = 0x002d (?) Failed to get btmesh model (0xbf42) attributes.
[E] Status: model_attr_status = 0x002d (?) Failed to get btmesh model (0xbf44) attributes.
Configuration of node (netkey_idx=0,addr=0x2005) is successful.

[I] Provisioning finished

Press enter to continue...
```

Ignore the failed models 0xbf42 and 0xbf44 as those are not Remote Provisioning-related.

Press ´Enter´ and check by pressing ´1´ if there is still one unprovisioned node and by pressing ´3´ if there is one provisioned node.

Press ´6´ to see available Remote Provisioning Servers. The one that was just provisioned should be displayed.

```
6
[I] Scanning for remote provisioner servers

[I] Remote provisioning server address: 0x2005

[I] Scanning for remote provisioner servers completed

Press enter to continue...
```

Press ´Enter´ and scan unprovisioned nodes using the Remote Provisioning Server by pressing ´7´.

```
7
Select a remote provisioning server
Type either the ID or the address from the list below

[I] 0. Remote provisioning server: 0x2005
```

Select the server by pressing ´0´.

```
0
[I] Remote scanning for unprovisioned nodes started

[I] Unprovisioned device UUID: 6e e9 34 56 5d 73 b5 5c aa 20 31 97 10 3b 4b 84

[I] Remote scanning completed

Press enter to continue...
```

You can see the Unprovisioned node via the remote server. Provision it by pressing ´Enter´ and then ´8´.

```
8
Select a remote provisioning server
Type either the ID or the address from the list below

[I] 0. Remote provisioning server: 0x2005
```

Press ´0´ to select the server.

```
0
[I] Selected remote provisioning server address 0x2005

Select the unprovisioned node for remote provisioning
Type either the ID, or UUID from the list below

[I] Unprovisioned node
[I] ID:      0
[I] UUID:    6e e9 34 56 5d 73 b5 5c aa 20 31 97 10 3b 4b 84
```

Press ´0´ to select the node.

```
0
[I] Remote provisioning...
[I] Device provisioned
[I] UUID:    6e e9 34 56 5d 73 b5 5c aa 20 31 97 10 3b 4b 84
[I] Address: 0x2008

Configuration of node (netkey_idx=0,addr=0x2008) is started.
DCD query of node (netkey_idx=0,addr=0x2008) completed.
Node (netkey_idx=0,addr=0x2008) runs btmesh_soc_light example.
[E] Status: model_attr_status = 0x002d (?) Failed to get btmesh model (0xbf42) attributes.
[E] Status: model_attr_status = 0x002d (?) Failed to get btmesh model (0xbf44) attributes.
Configuration of node (netkey_idx=0,addr=0x2008) is successful.

[I] Provisioning finished

Press enter to continue...
```

Ignore the failed models 0xbf42 and 0xbf44 as those are not Remote Provisioning related.

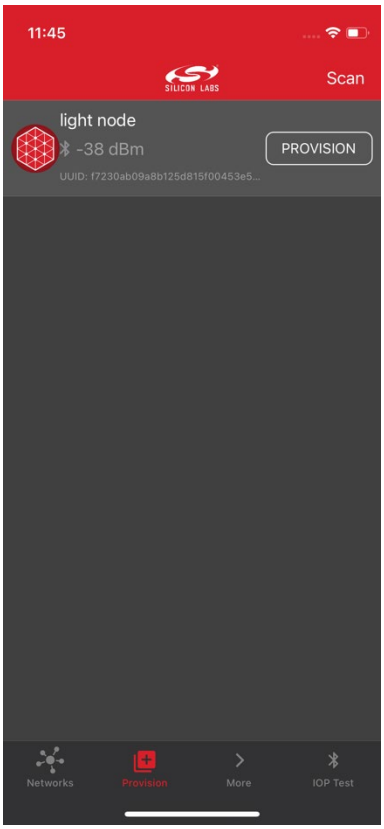
Press ´Enter´ and then press ´3´ to list the nodes in the network. Two should be available, the Remote Provisioning Server and the previously Unprovisioned node, remotely provisioned.

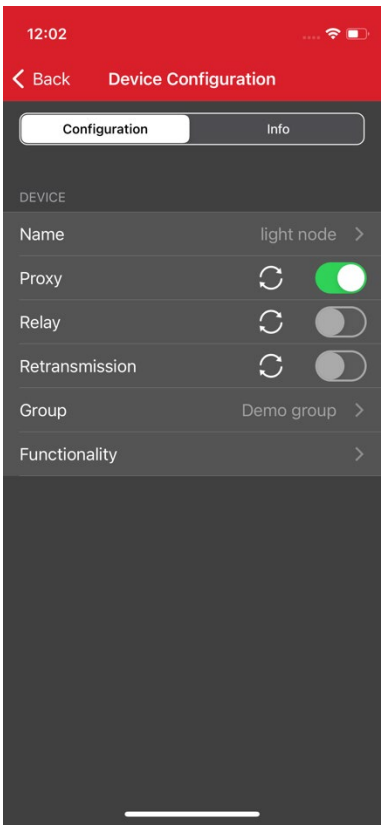
```
3
[I] Querying DDB list

[I] Address: 0x2005
[I] Element count: 3
[I] UUID:    6b 11 40 f7 be 9f e5 56 95 52 30 48 20 60 bc 69

[I] Address: 0x2008
[I] Element count: 3
[I] UUID:    6e e9 34 56 5d 73 b5 5c aa 20 31 97 10 3b 4b 84

Press enter to continue...
```

11.4 Running the Example with Smartphone App

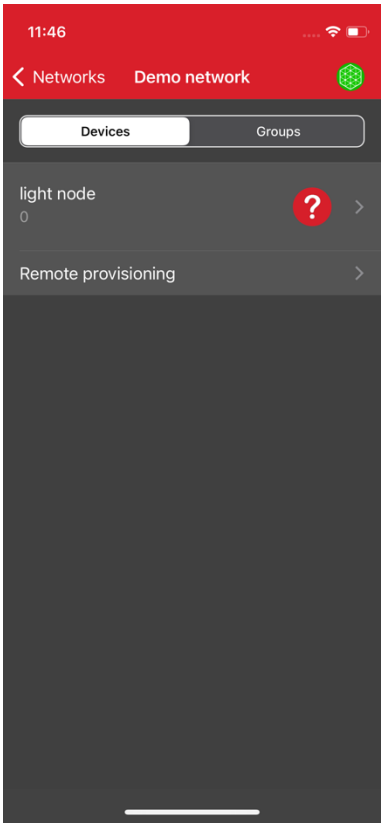
Install the Silicon Labs Bluetooth Mesh App from Google Play / App Store. When the “Remote Provisioning Server” and the “Unprovisioned Device” devices are up and running, start the provisioning process. To distinguish the two devices more easily, you can boot up the “Remote Provisioning Server” before the first provisioning.

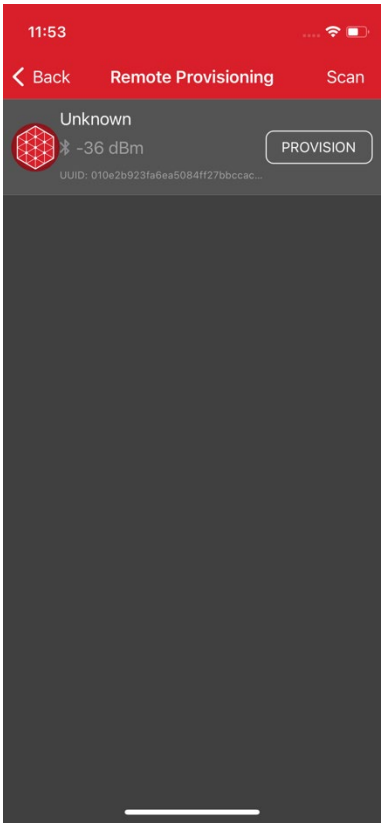
Scan for the “Remote Provisioning Server” device and provision it the usual way (turn on Proxy and add it to Group).



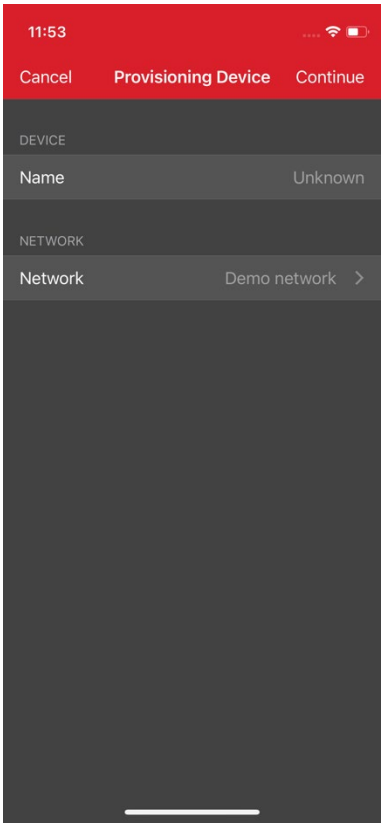


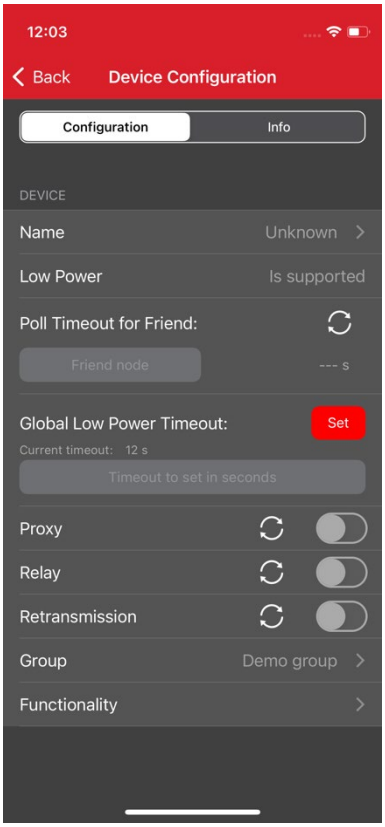
As the Remote Provisioning Server Model is added to the DCD, the App will offer the functionality in the network device list. By selecting it, you can scan for devices available for provisioning, via the Remote Provisioning Server (both devices should be online by now).



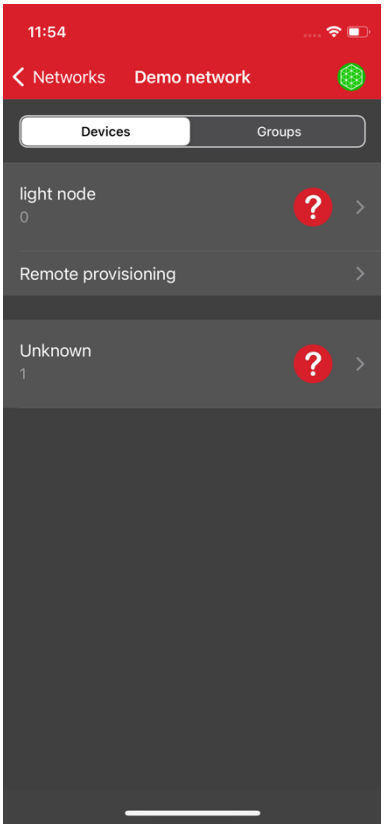


When the scanning found your “Unprovisioned Device”, you can provision and configure it the same way as you would with the standard provisioning.





If your efforts were successful, you will be able to see the newly added device in your network.



Simplicity Studio

One-click access to MCU and wireless tools, documentation, software, source code libraries & more. Available for Windows, Mac and Linux!

IoT Portfolio
www.silabs.com/IoT

SW/HW
www.silabs.com/simplicity

Quality
www.silabs.com/quality

Support & Community
www.silabs.com/community

SILICON LABS

Silicon Laboratories Inc.
400 West Cesar Chavez
Austin, TX 78701
USA

www.silabs.com