Step-by-Step Guide

How to Deploy BlueZ v5.50 on Raspberry Pi 3 and Use It

Part 2 — Provisioning

BlueZ is the official Linux *Bluetooth*® protocol stack. As stated in the BlueZ v5.47 release notes, “this release comes with initial support for it in the form of a new meshctl tool. Using this tool, it’s possible to provision mesh devices through the GATT Provisioning Bearer (PB-GATT), as well as communicate with them (e.g. configure them) using the GATT Proxy protocol.” This tutorial shows you how to build a new (unprovisioned) device, provisioned by meshctl on Raspberry Pi3 (R Pi3) board.

By the end of this step-by-step guide, you will be able to issue a meshctl command in the folder ~/bluez-5.50/mesh/, run the meshctl utility, and know how to use the meshctl utility to provision a new (unprovisioned) device and manage the network.

To learn the steps for installing BlueZ v5.50 on R Pi3, see Part 1 of this guide, Deployment.

Author: Kai Ren

Version: 1.0

Revision Date: 12 October 2018

Revision History

Version	Date	Author	Changes
1.0	2 October 2018	Kai Ren	Initial Version

table of contents

1.0 Prerequisite

Set Up the Development Environment

Before building a new device, you should follow this guide to setup the dev environment.

- Windows
- macOS
- Linux

Sample Project

This sample project illustrates how to use *meshctl* to provision and model config the device: this project is a sample from the Zephyr Project.

Please follow the sample project webpage to build and flash the firmware on the target board (nRF52840 DK). Then, open a serial terminal like Putty or Tera Term to monitor out-of-band (OOB) output from the target board. Select the correct serial port on your host computer, the serial terminal setting is:

- Baud rate: 115200
- Data bit: 8
- Parity: none
- Stop bit: 1

When you press the reset button on the board, the serial terminal should look like the image below.

For any other project which includes a *GenericOnOff Server* model and/or a *GenericOnOff Client* model, you can use this guide to provision and model config the device as well.

2.0 Provisioner Configuration

meshctl is a tool in *BlueZ v5.50* that works as a provisioner and distributes provisioning data (unicast address, NetKey, key index, and IV Index) to new, unprovisioned devices; with it, users can configure provisioning data by themselves. For example, change NetKey or IV Index, editing a json file on BlueZ can make it happen.

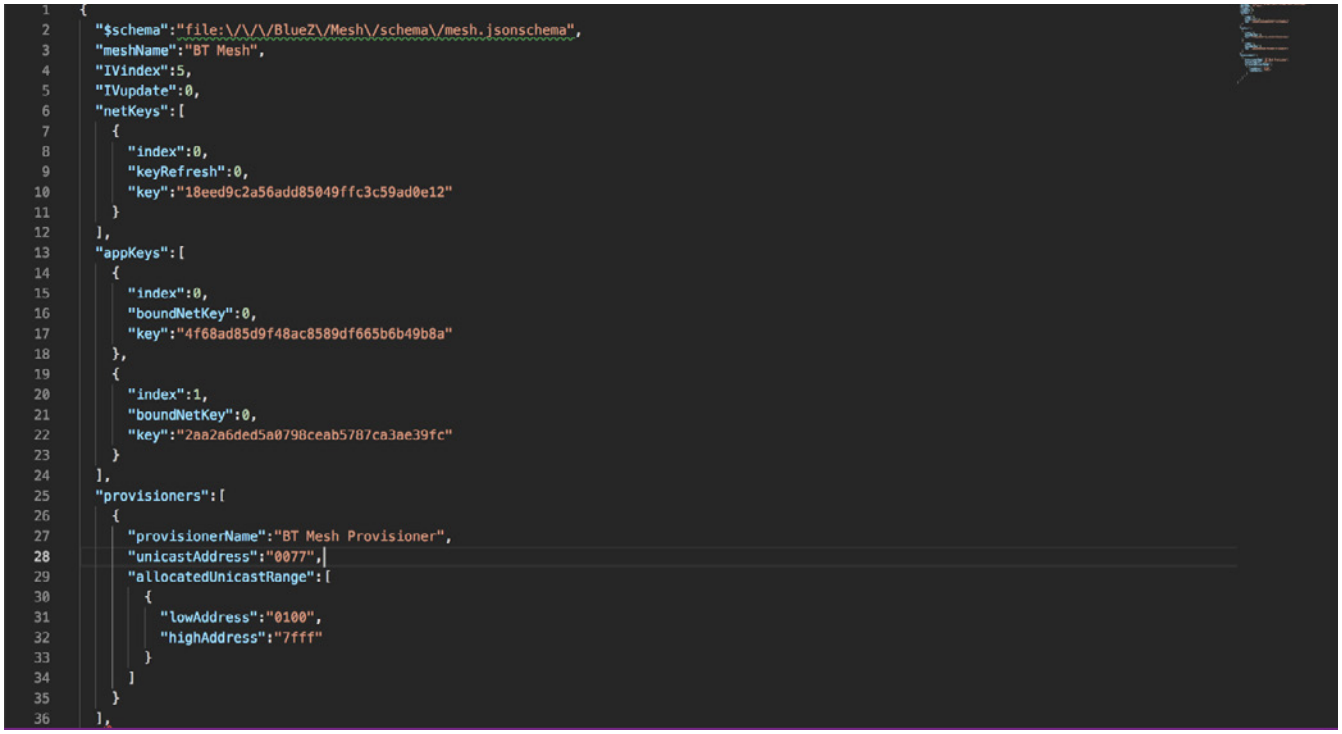
Open prov_db.json

Access the folder *./bluez-5.50/mesh/* and use the editor you would like to use to open *prov_db.json.* For this tutorial, we used *vi.*

```
pi@raspberrypi:~ $ cd bluez-5.50/mesh/
pi@raspberrypi:~/bluez-5.50/mesh $ vi prov_db.json
```

Edit prov_db.json

After opening *prov_db.json*, you should see:



If you want to change the *IV Index* setting, edit at line 4 as shown in the image below.

> “IVindex”: 5 #change IV Index, current setting is 5

If you want to change the *NetKey* setting, edit at line 6 as shown in the image below.

```
“netKeys”:[
  {
   “index”:0,   #change netkey index, current setting is 0
   “keyRefresh”:0,
   “key”:”18eed9c2a56add85049ffc3c59ad0e12”   #change netkey
  }
 ],
```

If you want to change the *AppKey* setting, edit at line 13 as shown in the image below.

```
“appKeys”:[
  {
   “index”:0,   #change AppKey index, current setting is 0
   “boundNetKey”:0,
   “key”:”F30AC76210160E03F2D8B4F1CF4510E2”#change AppKey0
  },
  {
   “index”:1,   #change AppKey index, current setting is 1
   “boundNetKey”:0,
   “key”:”F30AC76210160E03F2D8B4F1CF4510E2”#change AppKey1
  }
 ],
```

If you want to change the *Unicast Address* pool, edit at line 29 as shown in the image below.

```
“allocatedUnicastRange”:[
      {
        “lowAddress”:”0100”,    #head of unicast address poll
        “highAddress”:”7fff”     #tail of unicast address poll
      }
 ]
```

3.0 Provisioning

Commands Available on Main Menu

After reading Part 1, you should know how to launch *meshctl*. When *meshctl* is foreground, type *help* to list main menu commands as shown below.

```
pi@raspberrypi:~ $ cd bluez-5.50/mesh/
pi@raspberrypi:~/bluez-5.50/mesh $ meshctl
[meshctl]# help
Menu main:
Available commands:
-------------------
config                                  Configuration Model Submenu
onoff                                   On/Off Model Submenu
list                                    List available controllers
show [ctrl]                             Controller information
select <ctrl>                           Select default controller
security [0(low)/1(medium)/2(high)]     Display or change provision security level
info [dev]                              Device information
connect [net_idx] [dst]                 Connect to mesh network or node on network
discover-unprovisioned <on/off>         Look for devices to provision
provision <uuid>                        Initiate provisioning
power <on/off>                          Set controller power
disconnect [dev]                        Disconnect device
mesh-info                               Mesh networkinfo (provisioner)
local-info                              Local mesh node info
menu <name>                             Select submenu
version                                 Display version
quit                                    Quit program
exit                                    Quit program
help                                    Display help about this program
export                                  Print evironment variables
[meshctl]#
```

From the image above, you can see that every command has a description, and this guide shows you a few examples.

Main Menu Commands	Description
discover-unprovisioned <on/off>	Look for devices to provision
provision <uuid>	Initiate provisioning
disconnect <dev>	Disconnect device

Discover an Unprovisioned Device

To discover nearby unprovisioned devices, type in the command below.

```
discover-unprovisioned on
```

```
[meshctl]# discover-unprovisioned on
SetDiscoveryFilter success
Discovery started
Adapter property changed
[CHG] Controller 00:1B:DC:E0:38:29 Discovering: yes
                Mesh Provisioning Service (00001827-0000-1000-8000-00805f9b34fb)
                        Device UUID: 6bebf114000000000000000000000000
                        OOB: 0000
[NEW] Device ED:E4:14:F1:EB:6B ED-E4-14-F1-EB-6B
[meshctl]#
```

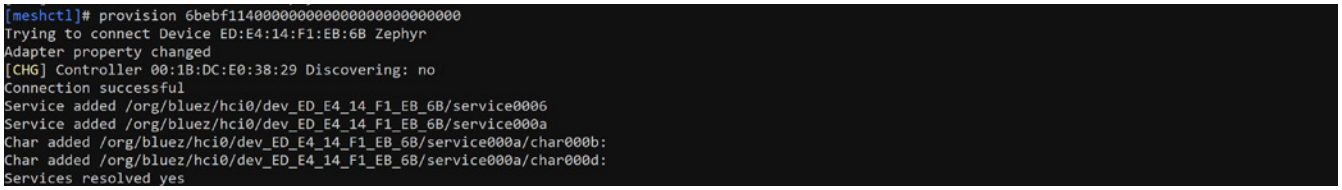
The device UUID is *6bebf1140000000000000000000000000*. The endian is LSB to MSB.

Provision

Copy the device UUID and paste it after the provision command, as shown below, to initiate the provision process.

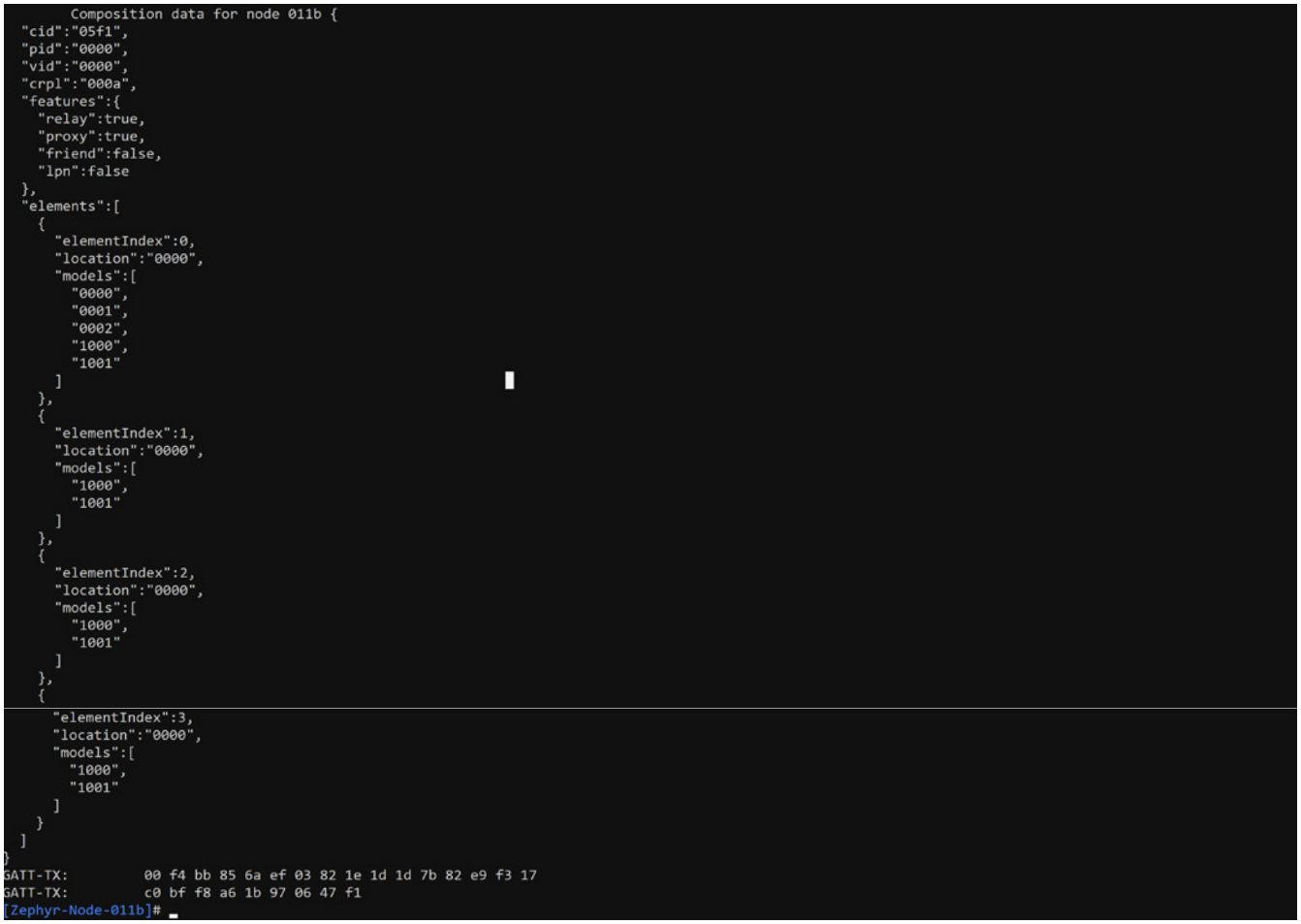
```
provision 6bebf1140000000000000000000000000   #paste the target device UUID on your own
```

After hitting the *Enter* key on your keyboard, you will see that the provisioning process is initiating, as shown in the image below.



Next, *meshctl* will ask you to type the OOB output value. This value can be obtained on the serial terminal. Please type the OOB output value as shown in the example below. You can see that the current OOB output value on serial terminal is *FSQR3G*.

If the OOB output value is correct, *meshctl* will move forward and reach the final step: *Composition Data* (details about *Composition Data* refer to Mesh Profile Specification v1.0, Section 4.2.1) as shown in the image below. This means the provisioning process was successful and the Provisioner got Composition Data from a just-provisioned node.



Composition

After interpreting the Composition Data, the information means:

- cid, company identifier of the node: 0×05f1
- pid, vendor-assigned product identifier of the node: 0×0000
- vid, vendor-assigned product version identifier of the node: 0×0000
- crpl, the minimum number of replay protection list entries in a device: 0×000a
- features:
    - relay: enabled;
    - proxy: enabled;
    - friend: disable;
    - low power node: disable

element	model	model id	unicast address (element address)	element index
Primary element	Configuration server	0×0000	0×011b	0
	Configuration client	0×0001		
	Health server	0×0002		
	GenericOnOff Server	0×1000		
	GenericOnOff Client	0×1001		
Secondary element	GenericOnOff Server	0×1000	0×011c	1
	GenericOnOff Client	0×1001		
Secondary element	GenericOnOff Server	0×1000	0×011d	2
Secondary element	GenericOnOff Server	0×1000	0×011e	3
	GenericOnOff Client	0×1001		

With the information above, you can start the model configuration.

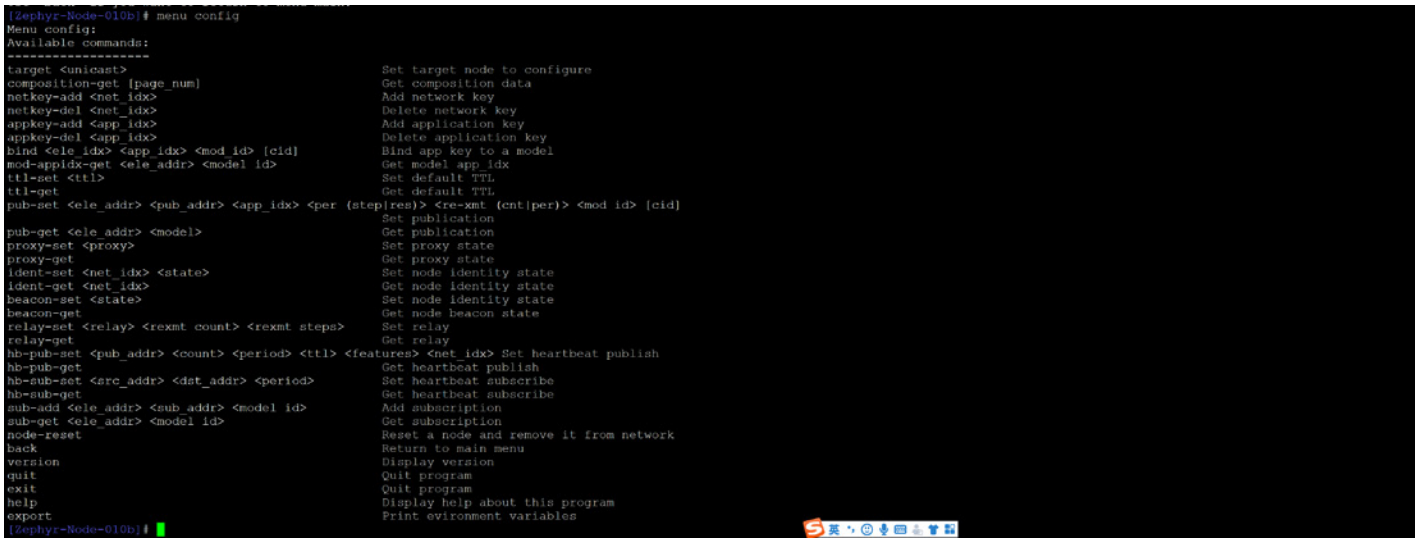
4.0 Model Configuration

Configuration

Now that provisioning is complete, it is time to perform model configuration. Type in the command below on main menu.

```
menu config
```

meshctl will go to the *menu config* submenu as shown in the image below.



The menu config screenshot above shows you how it supports the operations of Bluetooth mesh configuration and management like *NetKey, AppKey*, subscribe, publish, etc. Next, you will learn how to use some of these operations.

target 011b	# set primary element, 0×011b, to config
appkey-add 1	# add AppKey
bind 0 1 1000	# bind AppKey with certain model in certain element
sub-add 011b c000 1000	# add subscribe group address for certain model in certain element
bind 0 1 1001	# bind AppKey 1 to button 2 on element 1 (unicast 0101)
pub-set 010f c000 1 0 0 1001	# publish button 2 to group address c000

The table below explains the commands that were used.

Command	Description
target 011b	target <unicast> <unicast> -- unicast address Set the target node to configure, 011b is the unicast address of the primary element.
appkey-add 1	appkey-add <app_idx> <app_idx> -- AppKey index which points the key stored in *prov_db.json*, please refer to Edit prov_db.json.
bind 0 1 1000	bind <ele_idx> <app_idx> <mod_id> [cid] <ele_idx> -- element index, please refer to section Composition <app_idx> -- AppKey index which points the key stored in *prov_db.json* <mod id> -- model id, please refer to section Composition and Mesh Model Specification v1.0 <cid> -- optional
sub-add 010d c000 1000	sub-add <ele_addr> <sub_addr > <mod id> <ele_addr> -- element address a.k.a unicast address, please refer to section Composition <sub_addr> -- group address to subscribe, range: 0xc000 ~ 0xffff <mod id> -- model id, please refer to section Composition and Mesh Model Specification v1.0
pus-set 011b c000 1 0 0 1001	sub-add <ele_addr> <sub_addr > <mod id> <ele_addr> -- element address a.k.a unicast address, please refer to section Composition <sub_addr> -- group address to subscribe, range: 0xc000 ~ 0xffff <mod id> -- model id, please refer to section Composition and Mesh Model Specification v1.0

5.0 Verification

This section shows you how to use the command below to verify that the *GenericOnOffServer* model and *GenericOnOffClient* model are working.

Please type below commands on *meshctl*.

```
back
menu onoff
target 011b
onoff 1   #turn LED1 on
onoff 0  #turn LED1 off
get  #get LED1's status
```

Command	Description
back	Back to main menu.
menu onoff	Go to "menu onoff" menu.
target 011b	target <unicast> <unicast> -- unicast address Set the target node to configure, 011b is the unicast address of the primary element
onoff x	onoff <0/1> -- send "SET ON/OFF" command.
get	get – Get ON/OFF status.

Meanwhile, you also can use Button 1 on the board to control LED1 on or off: single click to turn.

6.0 Summary

After following these steps, you may find that there are 4 LEDs and 4 buttons on the board, as well as 4 elements which were assigned the unicast addresses from 0×011b to 0×011e, but here we just configure primary element, 0×011b. You should be able to configure the last three elements and models on your own.