Check your readiness for this session:

- **nRF Mesh** App. Haven’t installed it yet? Please scan…

iOS

Android
Bluetooth Mesh
Provisioning and Interoperability

Kai Ren, Senior Developer Relations Manager, Bluetooth SIG
Lighting

Air Conditioner

Station Occupancy

Sensor

Lighting
Build a Ceiling Light with Bluetooth mesh
Provisioner
Provisioning
Mesh system architecture

- Model Layer
- Foundation Model Layer
- Access Layer
- Upper Transport Layer
- Lower Transport Layer
- Network Layer
- Bearer Layer
- Bluetooth Low Energy Core Specification
Mesh system architecture

Model Layer
Foundation Model Layer
Access Layer
Upper Transport Layer
Lower Transport Layer
Network Layer
Bearer Layer
Bluetooth Low Energy Core Specification

Provisioning architecture

Provisioning Protocol
- Provisioning PDUs

Provisioning Transport
- Generic Provisioning PDU
- Proxy PDU
- PB-ADV
- PB-GATT

Provisioning Bearer
- Advertising
- Mesh Provisioning Service
Let Provisioner know: “I’m here, a new device nearby”.

Provisioner can receive the beacon and show it on the UI.

User has the right to select.
Invitation beaconing new device to report its provisioning capabilities including:
- the number of elements.
- security algorithms supported.
- the availability of its public key using OOB.
- OOB output action and size.
- OOB input action and size.
Provisioner selects a suitable provisioning method and inform New device.
Exchange public keys with each other.
\[ ECDH\text{Secret} = P-256(\text{private key, peer public key}) \]
Out-of-band, OOB communication is involved for secure device communication.

- Random number generated locally;
- Confirmation value is calculated by local random number, ECDHSecret, OOB information, etc..
- Exchange confirmation with each other
- Verify whether any MITM (Man-in-The-Middle) attack;
“Provisioning Data” includes:
- NetKey,
- Key index,
- Flags,
- IV Index,
- Unicast address of primary element address,

“Provisioning Data” is encrypted and distributed from Provisioner to Node.

Distribution

- Beaconing
- Invitation
- Exchange public keys
- Authentication
- Distribution of provisioning data
"Provisioning Data" includes:
- NetKey,
- Key index,
- Flags,
- IV Index,
- Unicast address of primary element address,

“Provisioning Data” is encrypted and distributed from Provisioner to Node.

Section 5.4.2
https://bit.ly/2NO1V8C
NetKey is a “seed” for different keys:
- Maintain a NetKey List;
- Up to 4096 keys in the list;
- Support key refresh;
NetKey is a “seed” for different keys:
- Maintain a NetKey List;
- Up 4096 keys in the list;
- Support key refresh;

Generate by “agreement”;
- A pair-wise key;
- Provisioner has all DevKey for each node;
- Be used to encrypt/decrypt foundation models messages;
Interoperability
Provisioner

Provisioning

Configuration
## Composition Data

<table>
<thead>
<tr>
<th>CID</th>
<th>PID</th>
<th>VID</th>
<th>CRPL</th>
<th>Features</th>
<th>Elements</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### Elements

<table>
<thead>
<tr>
<th>Loc</th>
<th>NumS</th>
<th>NumV</th>
<th>SIG Models</th>
<th>Loc</th>
<th>NumS</th>
<th>NumV</th>
<th>SIG Models</th>
<th>Vendor Models</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### SIG Model ID

<table>
<thead>
<tr>
<th>SIG Model ID</th>
<th>SIG Model ID</th>
<th>SIG Model ID</th>
<th>SIG Model ID</th>
<th>Vendor Model ID</th>
<th>Vendor Model ID</th>
</tr>
</thead>
</table>
element

MCU

Element Address

A A A

[Image of a circuit diagram with an MCU connected to three elements labeled A, A, A]
Unicast Address
Provisioner

Primary Element
- 0x0100
  - Secondary element 0x0101
  - Secondary element 0x0102

Primary Element
- 0x0103
  - Secondary element

Primary Element
- Secondary element

Primary Element
- Secondary element

Unicast Address
NetKey is a “seed” for different keys:
- Maintain a NetKey List;
- Up to 4096 keys in the list;
- Support key refresh;

- Generate by “agreement”;
- A pair-wise key;
- Provisioner has all DevKey for each node;
- Be used to encrypt/decrypt foundation models messages;

AppKey is added by Provisioner;
- Node maintains an AppKey list;
- Up to 4096 keys in the list;
- AppKey need to bind with model at certain element;
# Application Key

## AppKey List

<table>
<thead>
<tr>
<th>Index</th>
<th>AppKey</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x00</td>
<td>KEY0</td>
</tr>
<tr>
<td>0x01</td>
<td>KEY1</td>
</tr>
<tr>
<td>0x02</td>
<td>KEY2</td>
</tr>
<tr>
<td>0x03</td>
<td>KEY3</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>n</td>
<td>KEYn</td>
</tr>
</tbody>
</table>

## Node

- **Primary Element**
  - Configuration Server Model
  - Generic OnOff Server Model
  - Generic Level Server Model
  - Generic Battery Server Model
  - Vendor Model ...

- **Secondary Element**
  - Sensor Server Model
  - Light CTL Server Model
  - Light HSL Server Model
  - Light Lightness Server Model
  - Vendor Model ...

**AppKey & Model binding**
## Application Key

### AppKey List

<table>
<thead>
<tr>
<th>Index</th>
<th>AppKey</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x00</td>
<td>KEY0</td>
</tr>
<tr>
<td>0x01</td>
<td>KEY1</td>
</tr>
<tr>
<td>0x02</td>
<td>KEY2</td>
</tr>
<tr>
<td>0x03</td>
<td>KEY3</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>n</td>
<td>KEYn</td>
</tr>
</tbody>
</table>

### AppKey & Model binding

#### Primary Element
- Configuration Server Model
- Generic OnOff Server Model
- Generic Level Server Model
- Generic Battery Server Model
- Vendor Model ...

#### Secondary Element
- Sensor Server Model
- Light CTL Server Model
- Light HSL Server Model
- Light Lightness Server Model
- Vendor Model ...

---

26 May 2019
## Application Key

### AppKey List

<table>
<thead>
<tr>
<th>Index</th>
<th>AppKey</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x00</td>
<td>KEY0</td>
</tr>
<tr>
<td>0x01</td>
<td>KEY1</td>
</tr>
<tr>
<td>0x02</td>
<td>KEY2</td>
</tr>
<tr>
<td>0x03</td>
<td>KEY3</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>n</td>
<td>KEYn</td>
</tr>
</tbody>
</table>

### Node

- **Primary Element**
  - Configuration Server Model
  - Generic OnOff Server Model
  - Generic Level Server Model
  - Generic Battery Server Model
  - Vendor Model ...

- **Secondary Element**
  - Sensor Server Model
  - Light CTL Server Model
  - Light HSL Server Model
  - Light Lightness Server Model
  - Vendor Model ...

**AppKey & Model binding**
Application Key

<table>
<thead>
<tr>
<th>Index</th>
<th>AppKey</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x00</td>
<td>KEY0</td>
</tr>
<tr>
<td>0x01</td>
<td>KEY1</td>
</tr>
<tr>
<td>0x02</td>
<td>KEY2</td>
</tr>
<tr>
<td>0x03</td>
<td>KEY3</td>
</tr>
<tr>
<td></td>
<td>...</td>
</tr>
<tr>
<td>n</td>
<td>KEYn</td>
</tr>
</tbody>
</table>

AppKey List

Node

Primary Element
- Configuration Server Model
- Generic OnOff Server Model
- Generic Level Server Model
- Generic Battery Server Model
- Vendor Model ...

Secondary Element
- Sensor Server Model
- Light CTL Server Model
- Light HSL Server Model
- Light Lightness Server Model
- Vendor Model ...

AppKey & Model binding
## Application Key

### AppKey List

<table>
<thead>
<tr>
<th>Index</th>
<th>AppKey</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x00</td>
<td>KEY0</td>
</tr>
<tr>
<td>0x01</td>
<td>KEY1</td>
</tr>
<tr>
<td>0x02</td>
<td>KEY2</td>
</tr>
<tr>
<td>0x03</td>
<td>KEY3</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>n</td>
<td>KEYn</td>
</tr>
</tbody>
</table>

### AppKey & Model binding

#### Primary Element
- Configuration Server Model
- Generic OnOff Server Model
- Generic Level Server Model
- Generic Battery Server Model
- Vendor Model ...

#### Secondary Element
- Sensor Server Model
- Light CTL Server Model
- Light HSL Server Model
- Light Lightness Server Model
- Vendor Model ...

---

26 May 2019

Bluetooth SIG proprietary
## Unsegmented access message

<table>
<thead>
<tr>
<th>Preamble (4)</th>
<th>Access Address (4)</th>
<th>PDU</th>
<th>CRC (3)</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Adv PDU</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Header (2)</td>
<td>AdvA (6)</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>AD Structure</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>AD Len (1)</td>
<td>AD Type (1)</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>AD Data</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
### Unsegmented access message

<table>
<thead>
<tr>
<th>Preamble (4)</th>
<th>Access Address (4)</th>
<th>PDU</th>
<th>CRC (3)</th>
</tr>
</thead>
</table>

#### Adv PDU

<table>
<thead>
<tr>
<th>Header (2)</th>
<th>AdvA (6)</th>
<th>AD Structure</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>AD Len (1)</td>
<td>AD Type (1)</td>
</tr>
</tbody>
</table>

#### Network PDU

<table>
<thead>
<tr>
<th>IVI &amp; NID (1)</th>
<th>CTL &amp; TTL (1)</th>
<th>SEQ (3)</th>
<th>SRC (2)</th>
<th>DST (2)</th>
<th>TransportPDU</th>
<th>NetMIC 4/8</th>
</tr>
</thead>
</table>

### Unsegmented access message

<table>
<thead>
<tr>
<th><strong>Preamble (4)</strong></th>
<th><strong>Access Address (4)</strong></th>
<th><strong>Adv PDU</strong></th>
<th><strong>Network PDU</strong></th>
<th><strong>Lower Transport PDU</strong></th>
<th><strong>Upper Transport Access PDU</strong></th>
<th><strong>CRC (3)</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Header (2)</strong></td>
<td><strong>AdvA (6)</strong></td>
<td><strong>AD Structure</strong></td>
<td><strong>IVI &amp; NID (1)</strong></td>
<td><strong>CTL &amp; TTL (1)</strong></td>
<td><strong>SEQ (3)</strong></td>
<td><strong>SRC (2)</strong></td>
</tr>
<tr>
<td><strong>AD Len (1)</strong></td>
<td><strong>AD Type (1)</strong></td>
<td></td>
<td><strong>TransportPDU</strong></td>
<td><strong>NetMIC 4/8</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>AD Data</strong></td>
<td></td>
<td></td>
<td><strong>TransportPDU</strong></td>
<td><strong>NetMIC 4/8</strong></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**AD Structure**
- **AD Len (1)**
- **AD Type (1)**
- **AD Data**

**Network PDU**
- **IVI & NID (1)**
- **CTL & TTL (1)**
- **SEQ (3)**
- **SRC (2)**
- **DST (2)**
- **TransportPDU**
- **NetMIC 4/8**
## Unsegmented access message

<table>
<thead>
<tr>
<th>Preamble (4)</th>
<th>Access Address (4)</th>
<th>PDU</th>
<th>CRC (3)</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Adv PDU</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Header (2)</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>AdvA (6)</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>AD Len (1)</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>AD Type (1)</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>AD Structure</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>AD Data</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Network PDU</th>
<th>IVI &amp; NID (1)</th>
<th>CTL &amp; TTL (1)</th>
<th>SEQ (3)</th>
<th>SRC (2)</th>
<th>DST (2)</th>
<th>TransportPDU</th>
<th>NetMIC 4/8</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Lower Transport PDU</th>
<th>AKY &amp; AID (1)</th>
<th>Upper Transport Access PDU</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Upper Transport PDU</th>
<th>Encrypted payload (n)</th>
<th>Trans MIC (4/8)</th>
</tr>
</thead>
</table>
### Unsegmented access message

- **Preamble**: (4)
- **Access Address**: (4)
- **Adv PDU**: (2)
- **Header**: (6)
- **AdvA**: (6)
- **PDU**: (3)
- **CRC**: (3)

#### Adv PDU
- **Header**: (2)
- **AdvA**: (6)

#### AD Structure
- **AD Len**: (1)
- **AD Type**: (1)
- **AD Data**: (2)

#### Network PDU
- **IVI & NID**: (1)
- **CTL & TTL**: (1)
- **SEQ**: (3)
- **SRC**: (2)
- **DST**: (2)
- **TransportPDU**: (2)
- **NetMIC**: 4/8

#### Lower Transport PDU
- **AKY & AID**: (1)

#### Upper Transport PDU
- **Encrypted payload**: (n)
- **Trans MIC**: 4/8

#### Access message
- **opcode**: (n)
- **parameter**: (n)

**Opcode and parameter defined in Mesh Model Specification 1.0**
**Unsegmented access message**

<table>
<thead>
<tr>
<th>Preamble</th>
<th>Access Address</th>
<th>PDU</th>
<th>CRC</th>
</tr>
</thead>
<tbody>
<tr>
<td>(4)</td>
<td>(4)</td>
<td></td>
<td>(3)</td>
</tr>
</tbody>
</table>

**Adv PDU**

**Header** (2)  
**AdvA** (6)  

**AD Structure**

<table>
<thead>
<tr>
<th>AD Len</th>
<th>AD Type</th>
<th>AD Data</th>
</tr>
</thead>
<tbody>
<tr>
<td>(1)</td>
<td>(1)</td>
<td></td>
</tr>
</tbody>
</table>

**Network PDU**

<table>
<thead>
<tr>
<th>IVI &amp; NID</th>
<th>CTL &amp; TTL</th>
<th>SEQ</th>
<th>SRC</th>
<th>DST</th>
<th>TransportPDU</th>
<th>NetMIC 4/8</th>
</tr>
</thead>
<tbody>
<tr>
<td>(1)</td>
<td>(1)</td>
<td>(3)</td>
<td>(2)</td>
<td>(2)</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Lower Transport PDU**

**Upper Transport Access PDU**

**Upper Transport PDU**

**Encrypted payload** (n)  
**Trans MIC 4/8**

**Opcode and parameter defined in Mesh Model Specification 1.0**

**Access message**

<table>
<thead>
<tr>
<th>op code</th>
<th>parameter</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>(n)</td>
</tr>
</tbody>
</table>
### Unsegmented access message

<table>
<thead>
<tr>
<th>Preamble (4)</th>
<th>Access Address (4)</th>
<th><strong>PDU</strong></th>
<th><strong>CRC</strong> (3)</th>
</tr>
</thead>
</table>

#### Adv PDU

<table>
<thead>
<tr>
<th>Header (2)</th>
<th>AdvA (6)</th>
<th><strong>AD Structure</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>AD Len</strong> (1)</td>
<td><strong>AD Type</strong> (1)</td>
<td><strong>AD Data</strong></td>
</tr>
</tbody>
</table>

#### Network PDU

<table>
<thead>
<tr>
<th>IVI &amp; NID (1)</th>
<th>CTL &amp; TTL (1)</th>
<th>SEQ (3)</th>
<th>SRC (2)</th>
<th>DST (2)</th>
<th>TransportPDU</th>
<th><strong>NetMIC 4/8</strong></th>
</tr>
</thead>
</table>

#### Lower Transport PDU

<table>
<thead>
<tr>
<th><strong>AKY &amp; AID</strong> (1)</th>
<th><strong>Upper Transport Access PDU</strong></th>
</tr>
</thead>
</table>

#### Upper Transport PDU

<table>
<thead>
<tr>
<th><strong>Encrypted payload</strong> (n)</th>
<th><strong>Trans MIC 4/8</strong></th>
</tr>
</thead>
</table>

**Opcode and parameter defined in Mesh Model Specification 1.0**

Access message

- op code
- parameter (n)
Unsegmented access message

Preamble (4)  Access Address (4)  PDU  CRC (3)

<table>
<thead>
<tr>
<th>PDU</th>
<th>Header (2)</th>
<th>AdvA (6)</th>
<th>AD Structure</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>AD Len (1)</td>
<td>AD Type (1)</td>
<td>AD Data</td>
</tr>
<tr>
<td></td>
<td>PrivacyKey obfuscated</td>
<td>EncryptionKey encrypted</td>
<td></td>
</tr>
</tbody>
</table>

Network PDU

<table>
<thead>
<tr>
<th>IVI &amp; NID (1)</th>
<th>CTL &amp; TTL (1)</th>
<th>SEQ (3)</th>
<th>SRC (2)</th>
<th>DST (2)</th>
<th>TransportPDU</th>
<th>NetMIC 4/8</th>
</tr>
</thead>
</table>

Lower Transport PDU

<table>
<thead>
<tr>
<th>AKY &amp; AID (1)</th>
<th>Upper Transport Access PDU</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>AppKey/DevKey encrypted</td>
</tr>
</tbody>
</table>

Upper Transport PDU

<table>
<thead>
<tr>
<th>Opcode and parameter defined in Mesh Model Specification 1.0</th>
</tr>
</thead>
</table>

Access message

<table>
<thead>
<tr>
<th>op code</th>
<th>parameter (n)</th>
</tr>
</thead>
</table>

Encrypted payload (n)

Trans MIC 4/8
Hands-on
Hands-on

- Open the bag, take micro:bit board out;
- Connect board to your computer by micro USB cable, a new volume naming “MICROBIT” appears;
- Take USB Disk out and connect to computer;
- Copy BluetoothMeshFW.hex* and paste it into “MICROBIT” volume;
- Source code is available here;

* Provisioning and model configuration will be erased after a reset;
<table>
<thead>
<tr>
<th>Provisioner</th>
<th>Kit</th>
<th>Stack</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Apple Watch 3</strong></td>
<td><strong>Nordic</strong></td>
<td><strong>Zephyr v1.14.0</strong></td>
</tr>
<tr>
<td></td>
<td>nRF52840 Dev Kit, PCA10056</td>
<td></td>
</tr>
<tr>
<td><strong>iPhone 8</strong></td>
<td><strong>Silicon Labs</strong></td>
<td><strong>Bluetooth mesh v1.2.0</strong></td>
</tr>
<tr>
<td></td>
<td>EFR32 Blue Gecko Bluetooth Starter Kit</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Cypress CYW920719Q40EVB-01</td>
<td><strong>WICED 6.20</strong></td>
</tr>
<tr>
<td><strong>Pixel 2</strong></td>
<td><strong>STMicroelectronics</strong></td>
<td><strong>BlueNRG-Mesh V1.06.00</strong></td>
</tr>
<tr>
<td></td>
<td>STEVAL-IDB008V2</td>
<td></td>
</tr>
<tr>
<td></td>
<td><strong>Micro:bit Education Foundation</strong></td>
<td><strong>Zephyr v1.14.0</strong></td>
</tr>
<tr>
<td></td>
<td>micro:bit board</td>
<td></td>
</tr>
</tbody>
</table>

Blog: A Developer’s Guide for Proving Bluetooth Mesh Interoperability
Step-by-Step Guide

How to Deploy BlueZ v5.49 on Raspberry Pi3 and Use It

Part 1

BlueZ is the official Linux Bluetooth® protocol stack. As stated in the BlueZ v5.49 release notes, "This release centers 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 guides you through the steps for installing BlueZ v5.49 on Raspberry Pi3 (R Pi3).

Author: Kai Ren
Version: 1.0
Revision Date: 31 May 2019

Now available at https://bit.ly/2JFFDaN
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/meshctl, 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

Thank you!

谢谢