Check your readiness for this session:

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

iOS

Android

#BluetoothAsia2019#

BluetoothAsia
探索、创新、开拓

Bluetooth Mesh Provisioning and Interoperability

Kai Ren, Senior Developer Relations Manager, Bluetooth SIG

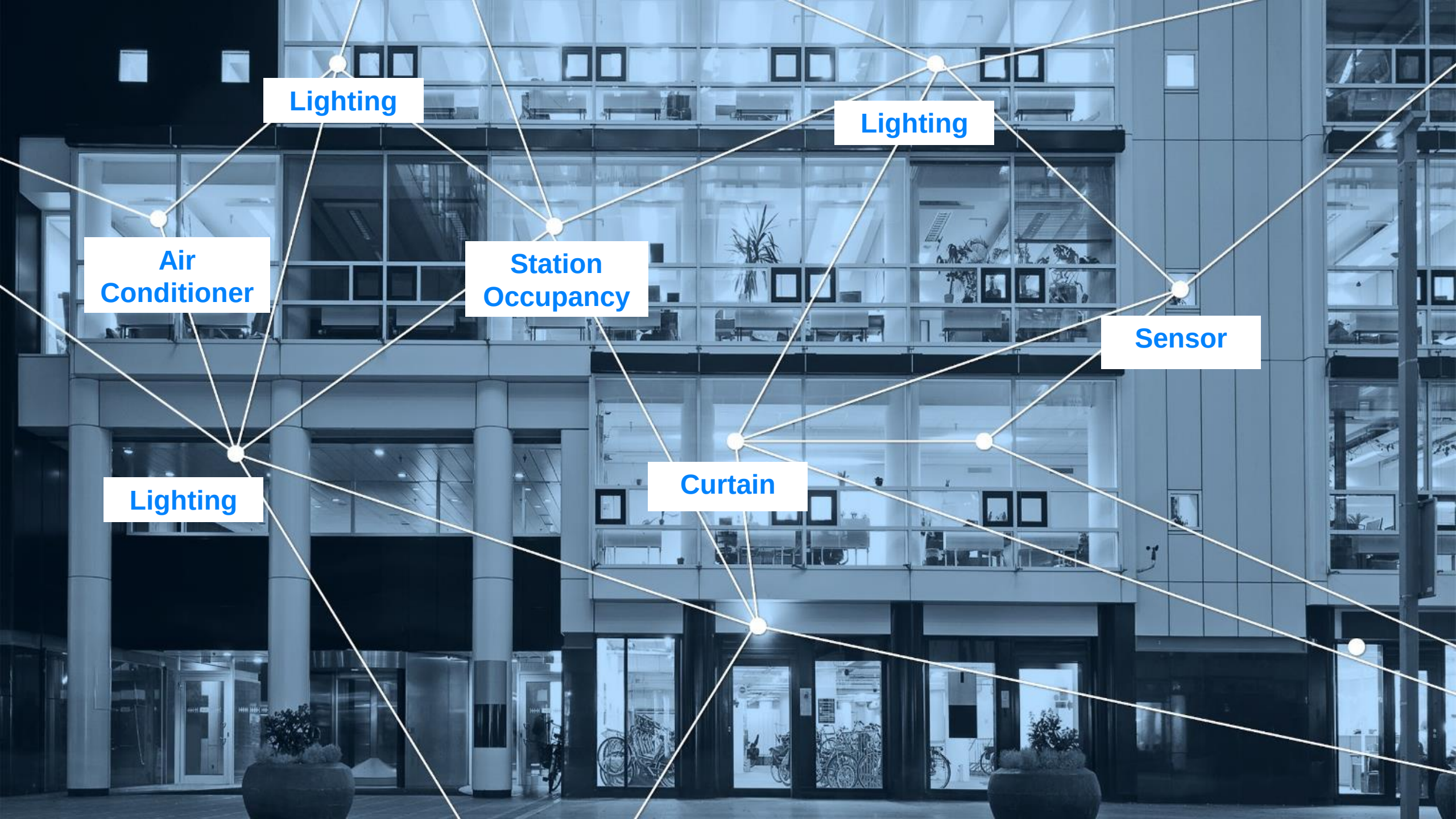
微信

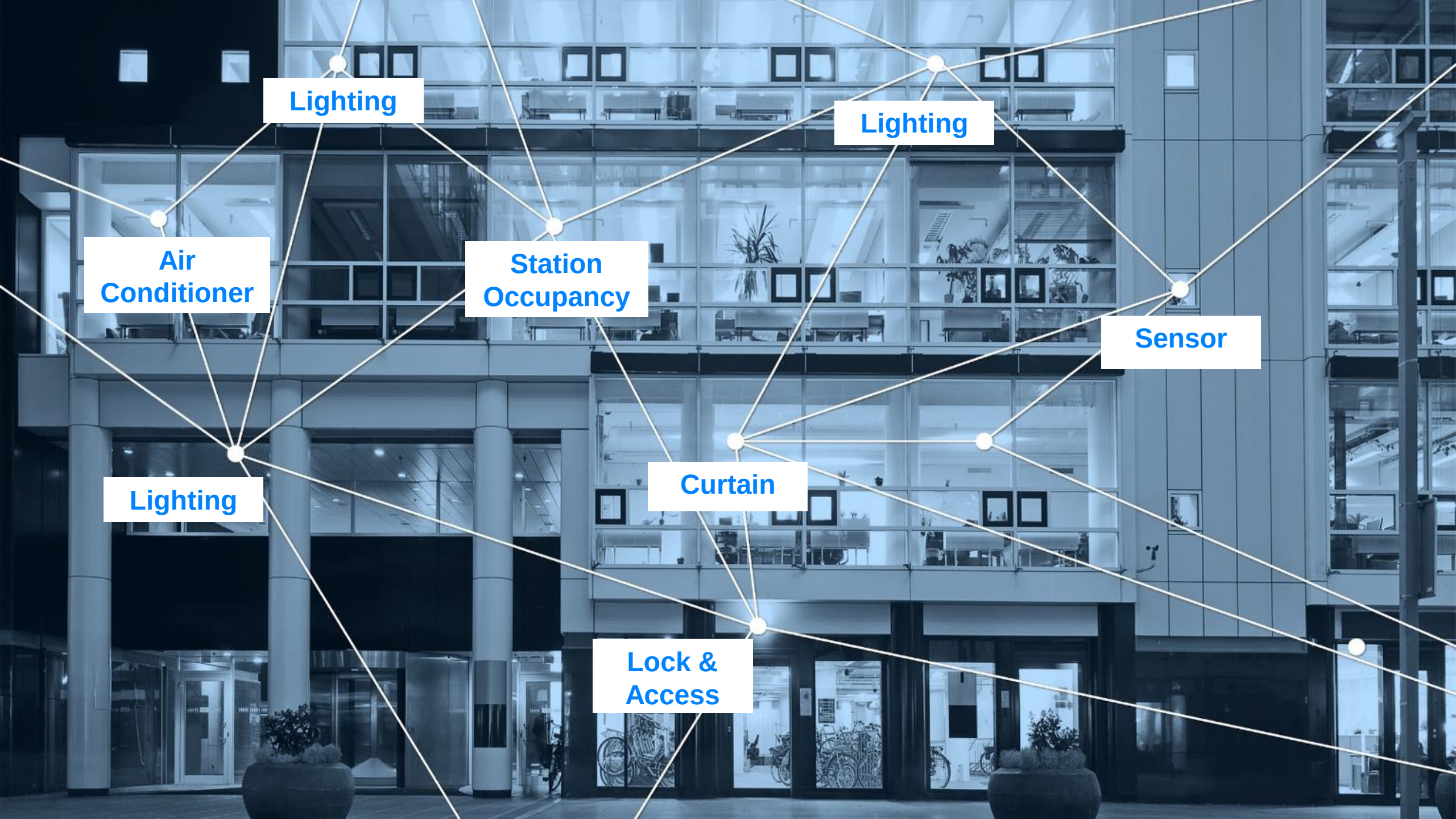

Lighting
Lighting
Lighting


Lighting
Lighting
Air Conditioner
Lighting

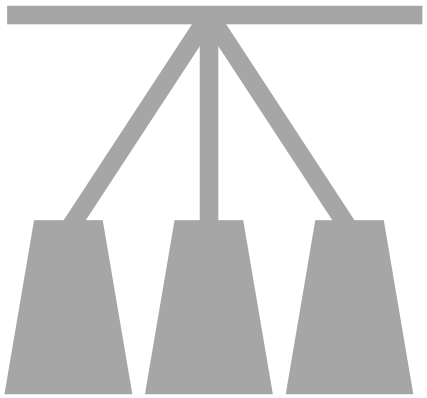

Lighting
Lighting
Air Conditioner
Station Occupancy
Lighting

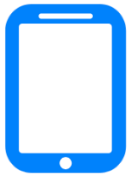

Lighting
Lighting
Air Conditioner
Station Occupancy
Sensor
Lighting


Lighting
Lighting
Air Conditioner
Station Occupancy
Sensor
Lighting
Curtain

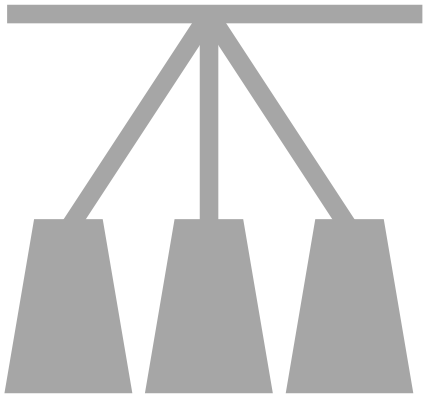

Lighting
Lighting
Air Conditioner
Station Occupancy
Sensor
Lighting
Curtain
Lock & Access

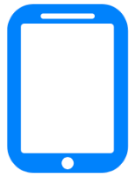
Build a Ceiling Light with Bluetooth mesh



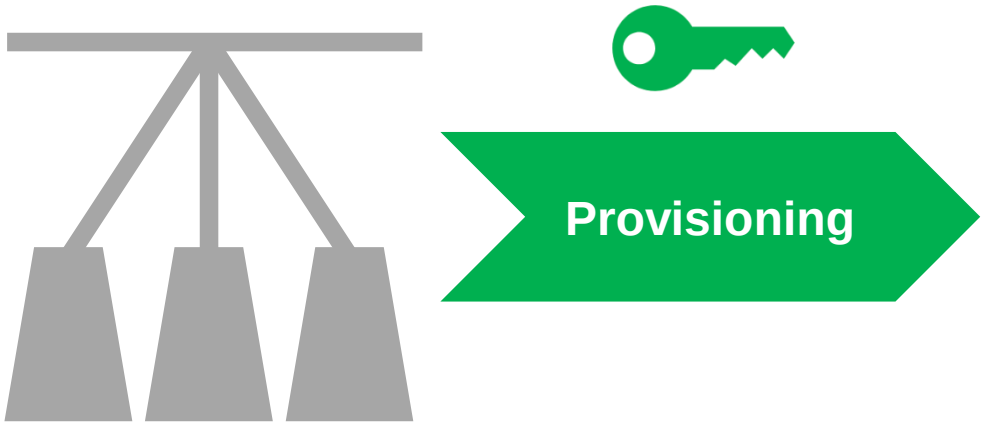


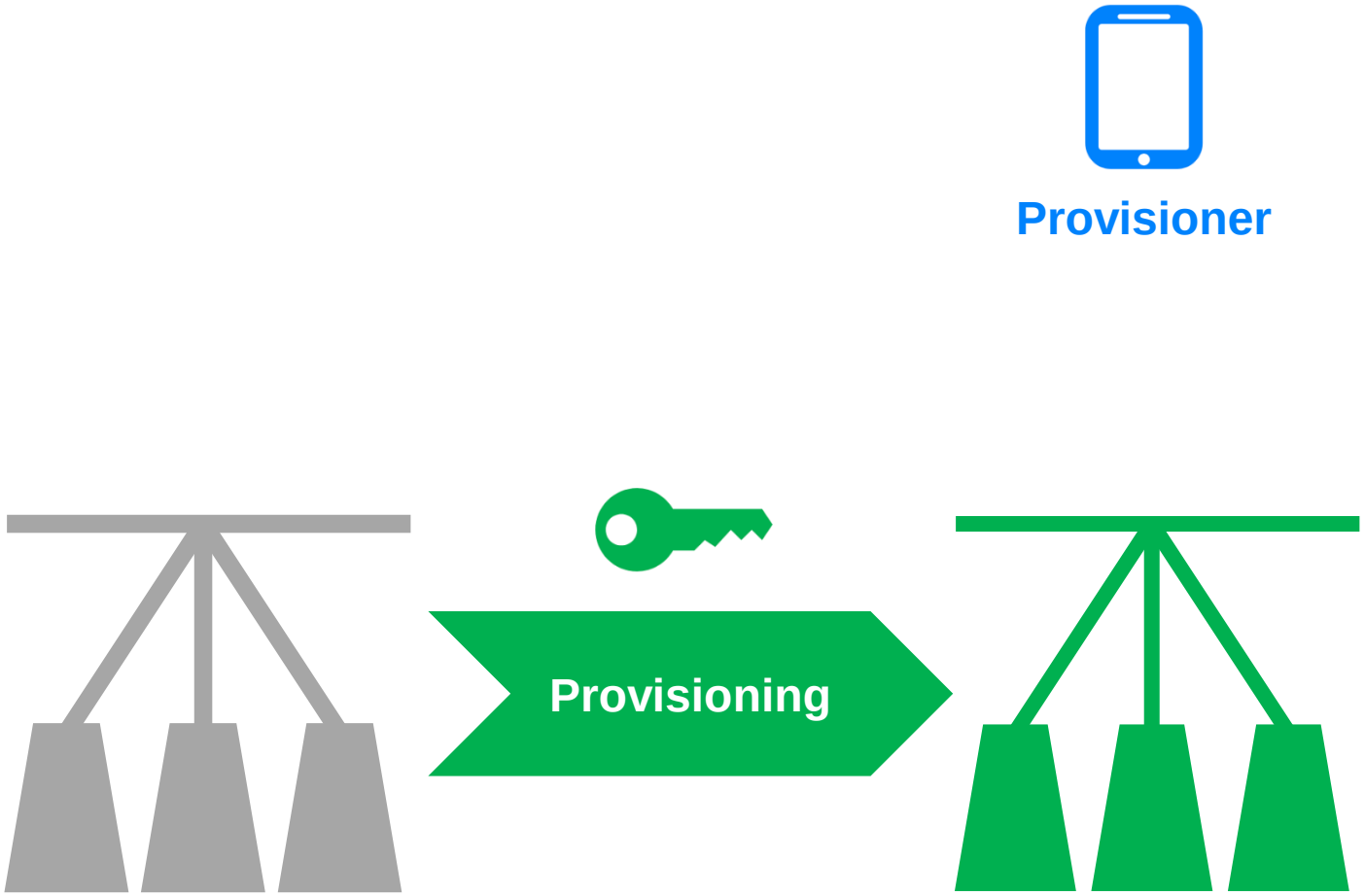
Provisioner

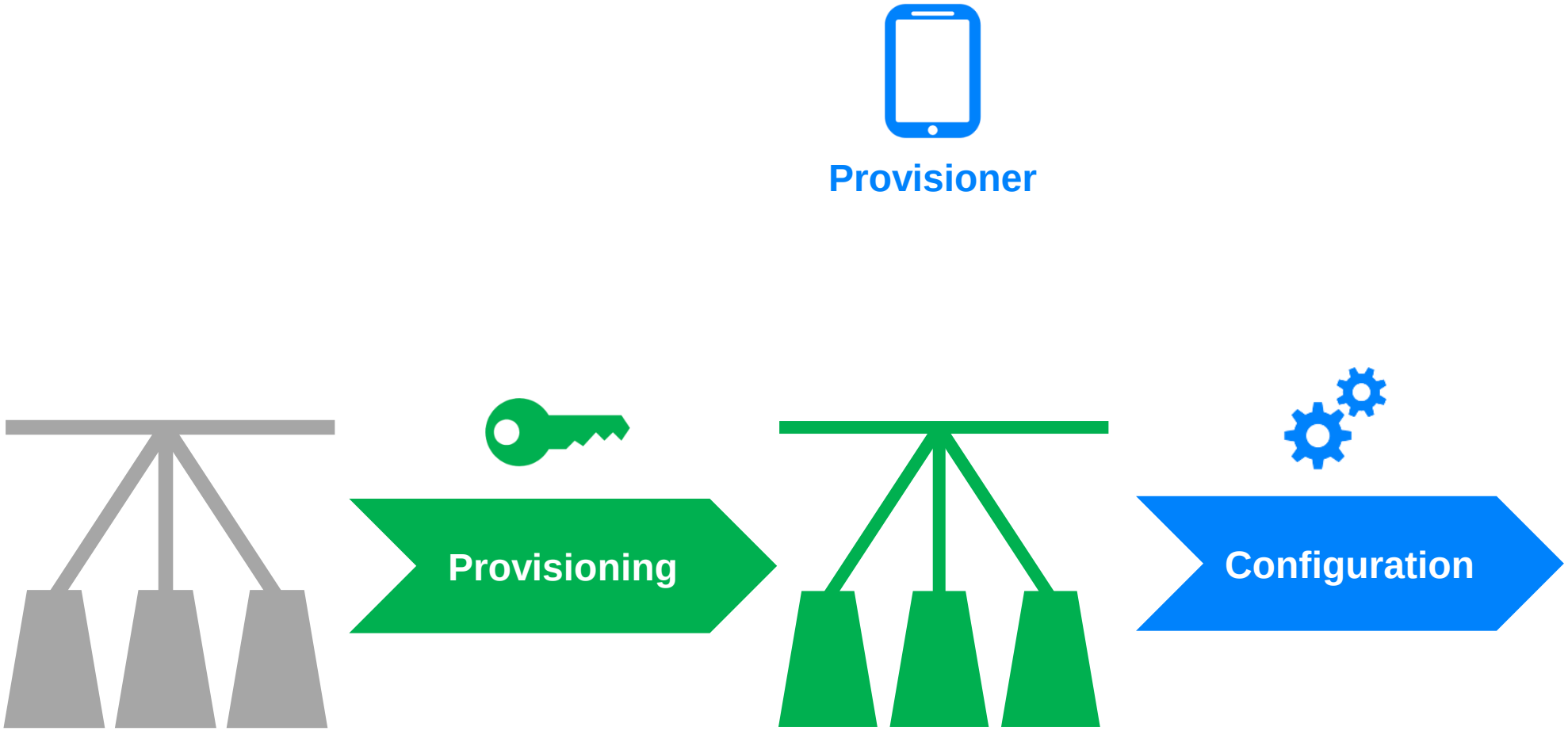


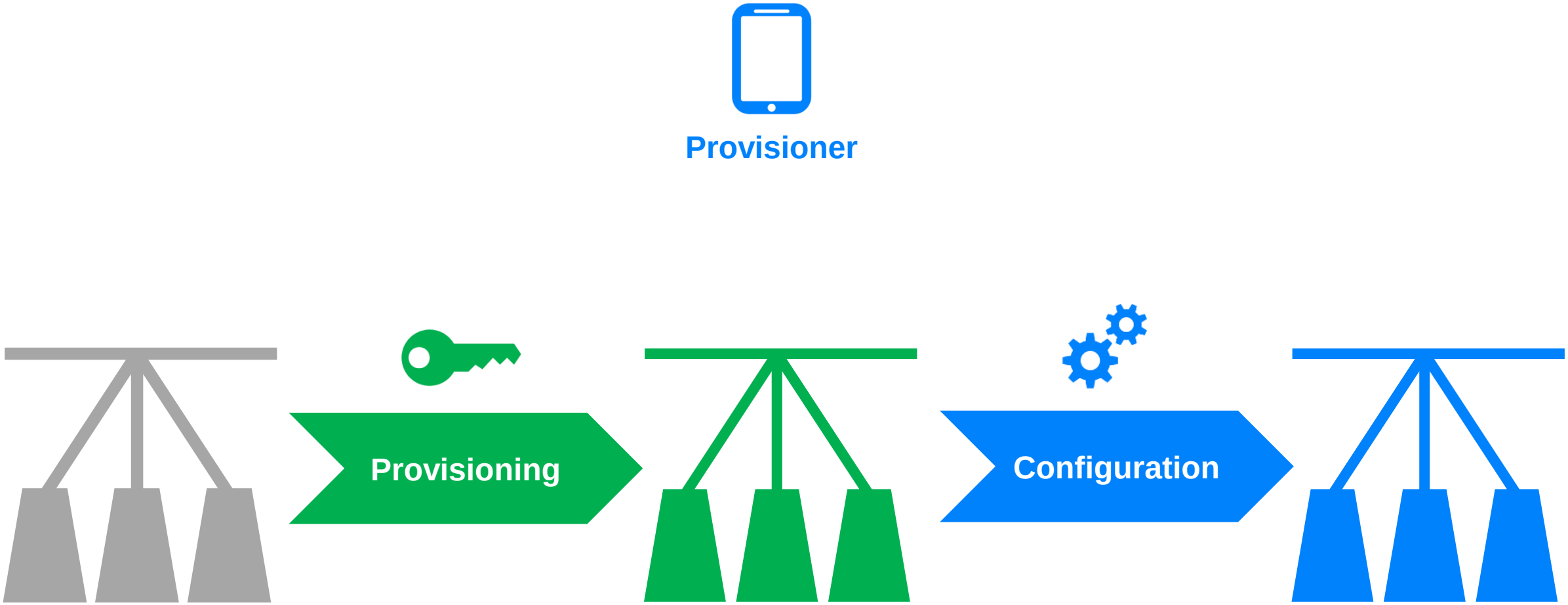


Provisioner



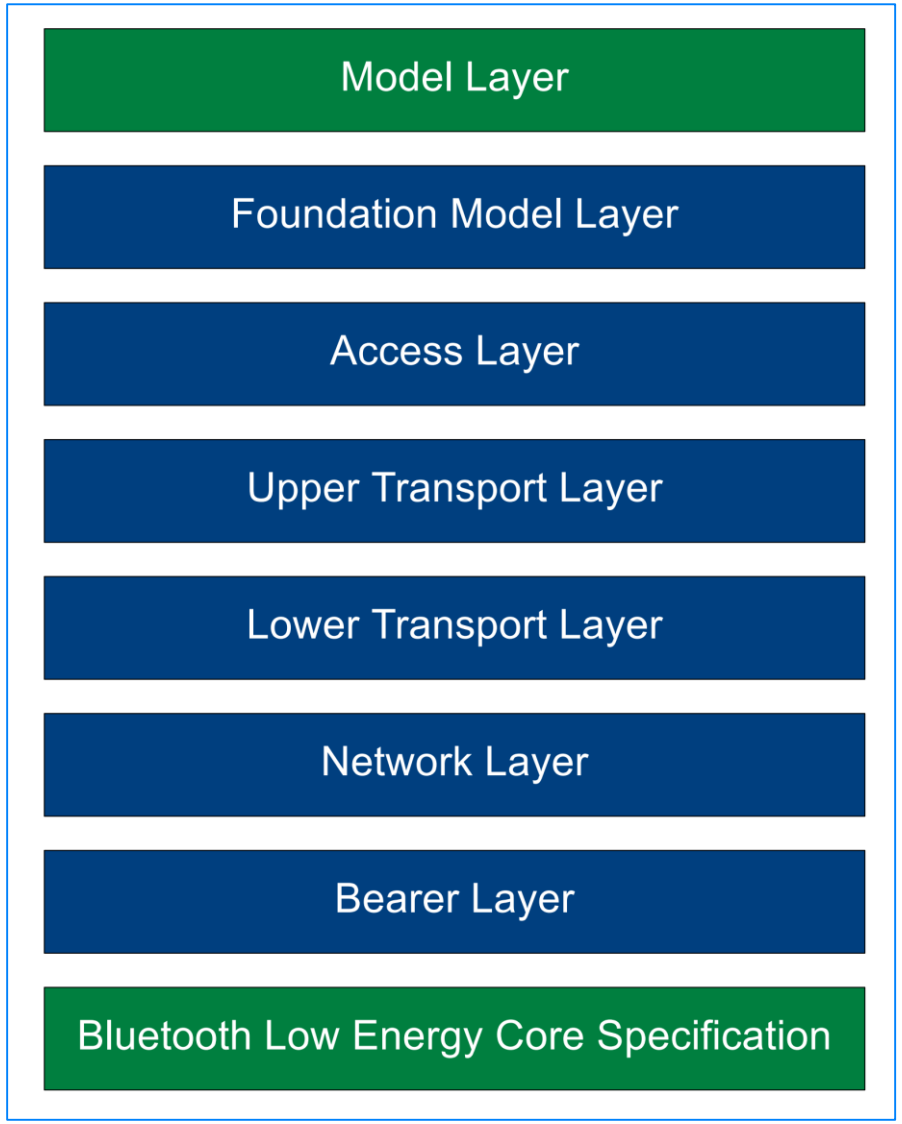

Provisioner
Provisioning


Provisioner
Provisioning
Configuration


Provisioner
Provisioning
Configuration

Provisioning

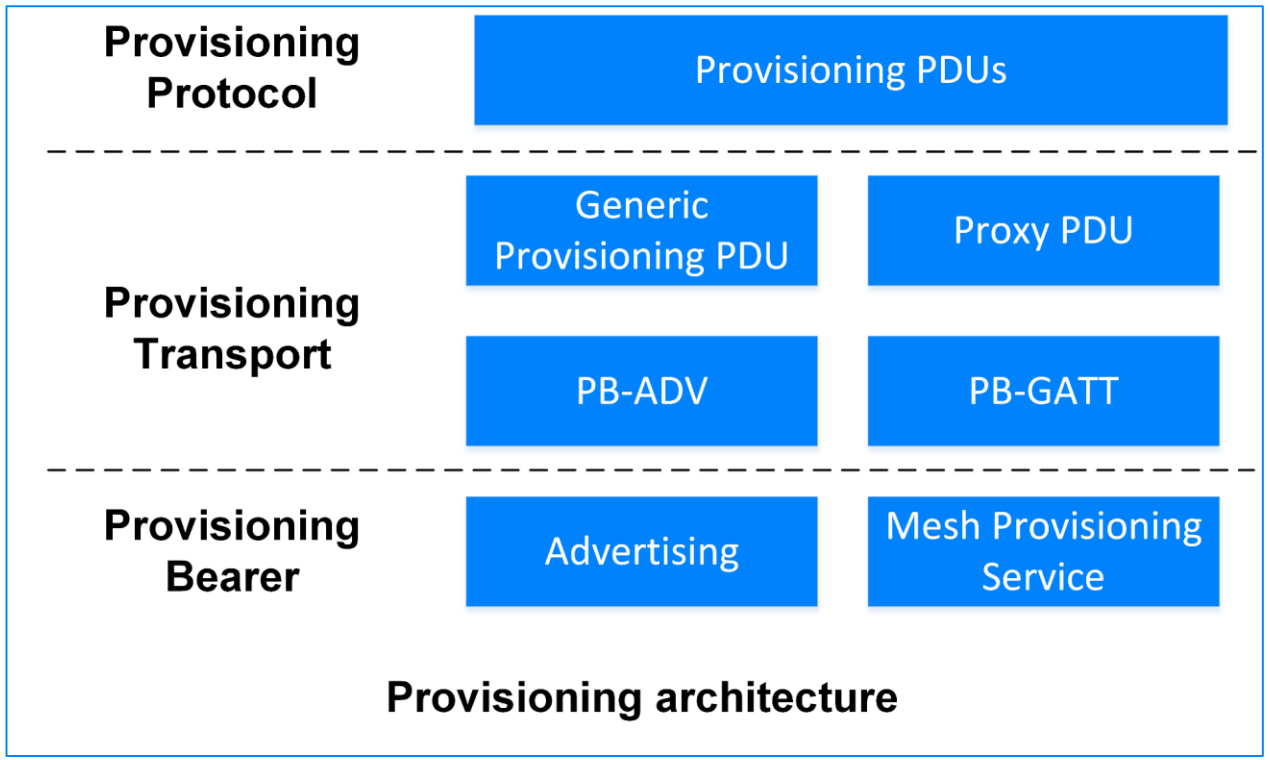
provisioning



Mesh system architecture

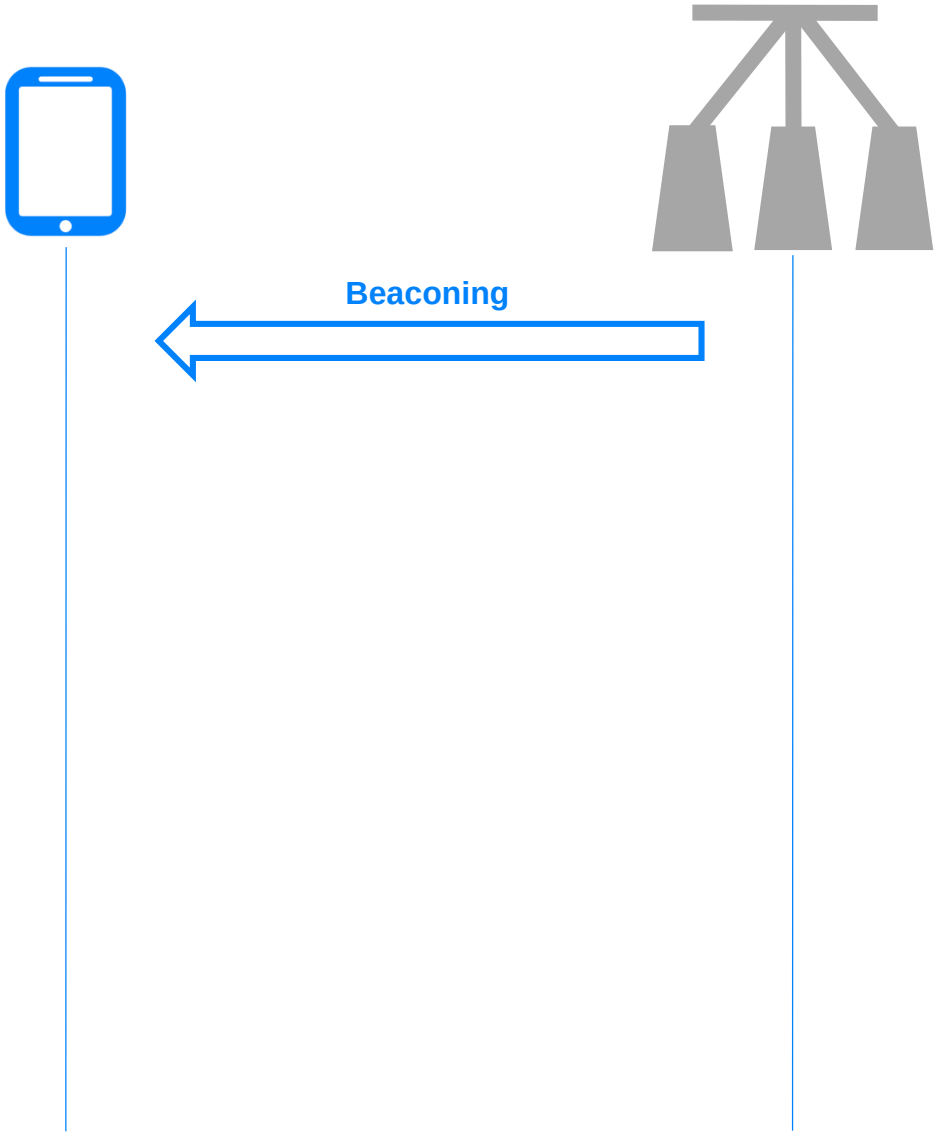
provisioning

Mesh system architecture



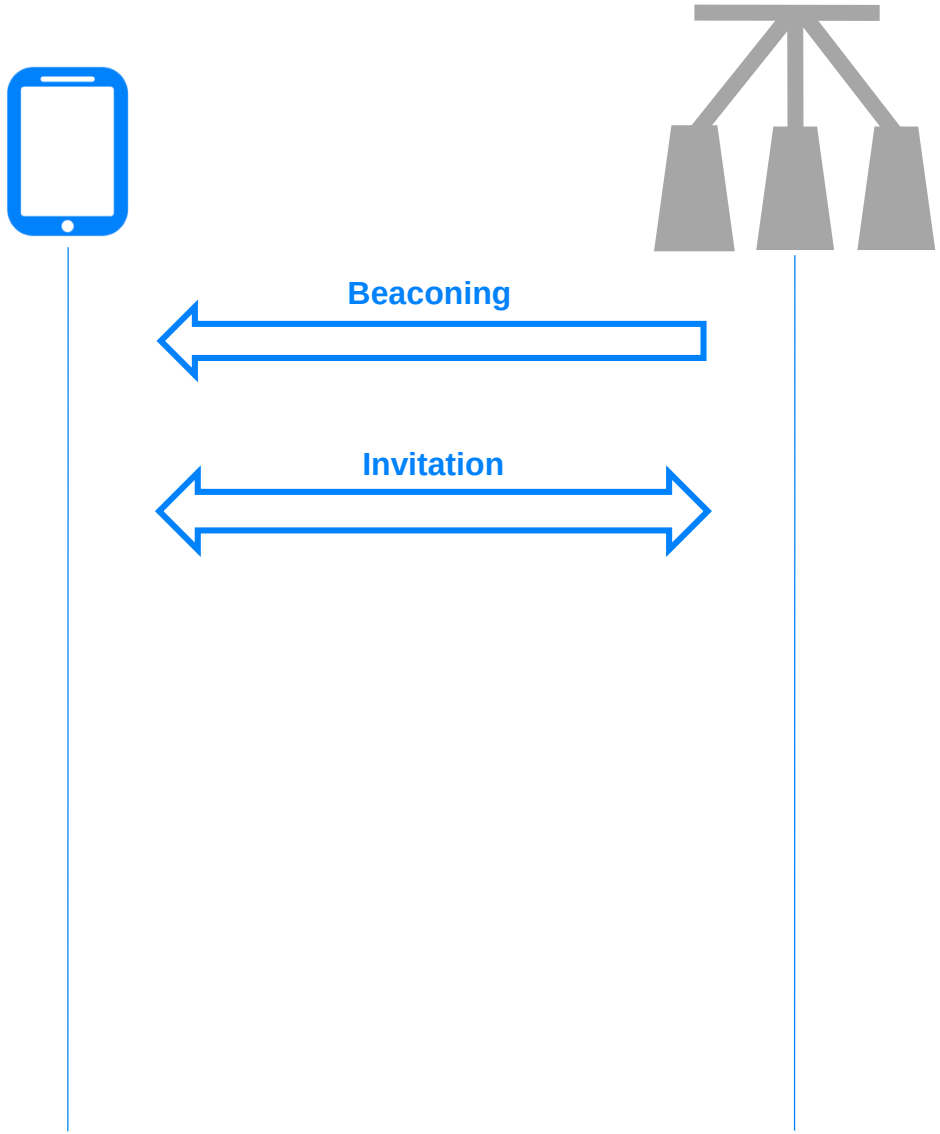
Provisioning architecture

Beaconing



- 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.

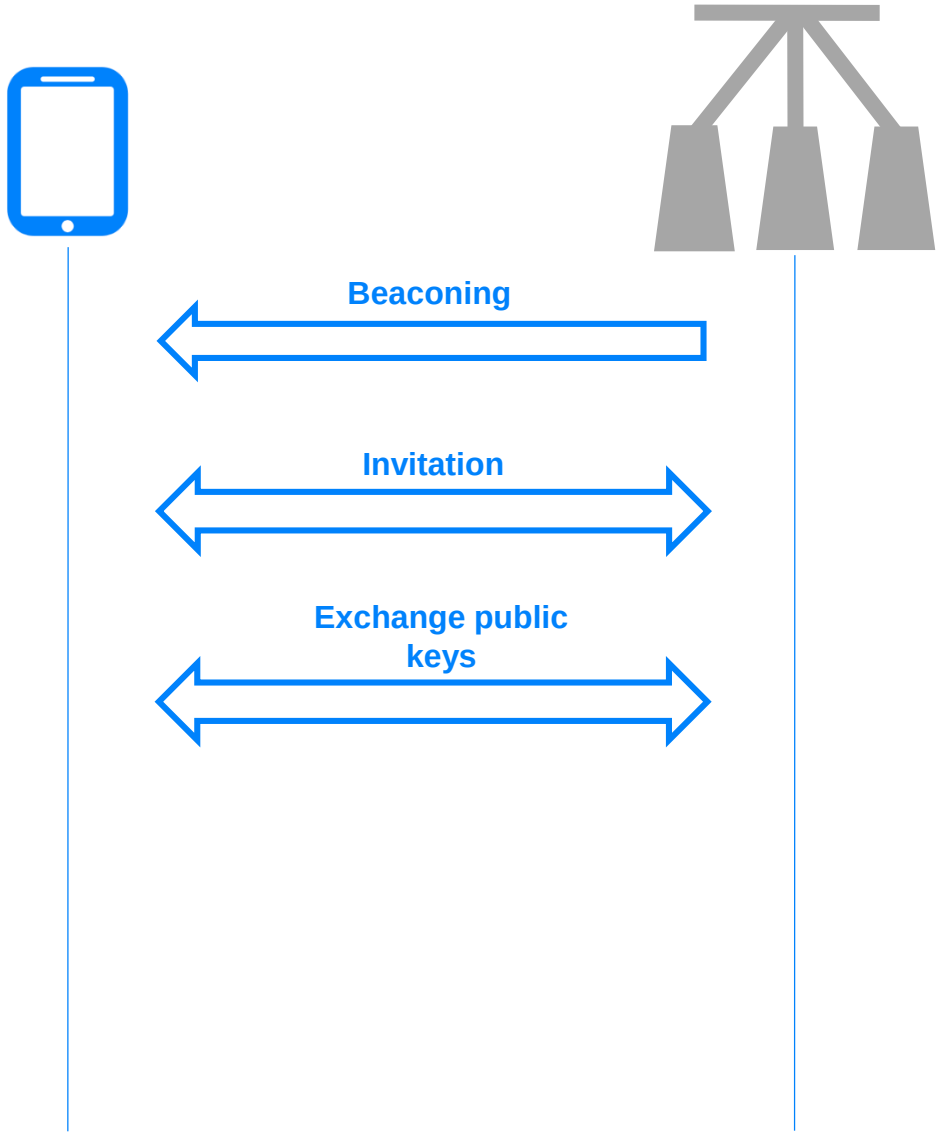
Invitation



New device to report its provisioning capabilities including:

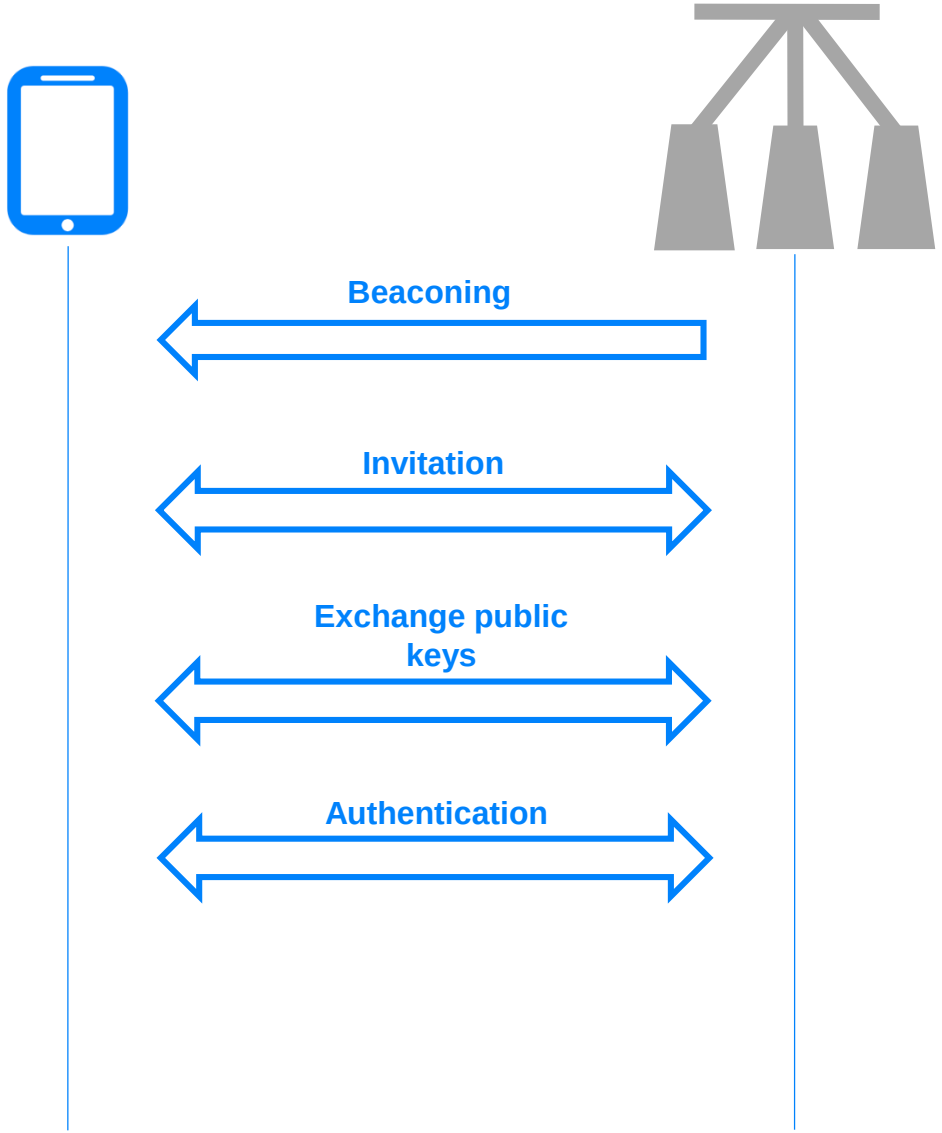
- ❑ the number of elements.
- ❑ security algorithms supported.
- ❑ the availability of its public key using OOB.
- ❑ OOB output action and size.
- ❑ OOB input action and size.

Public Key



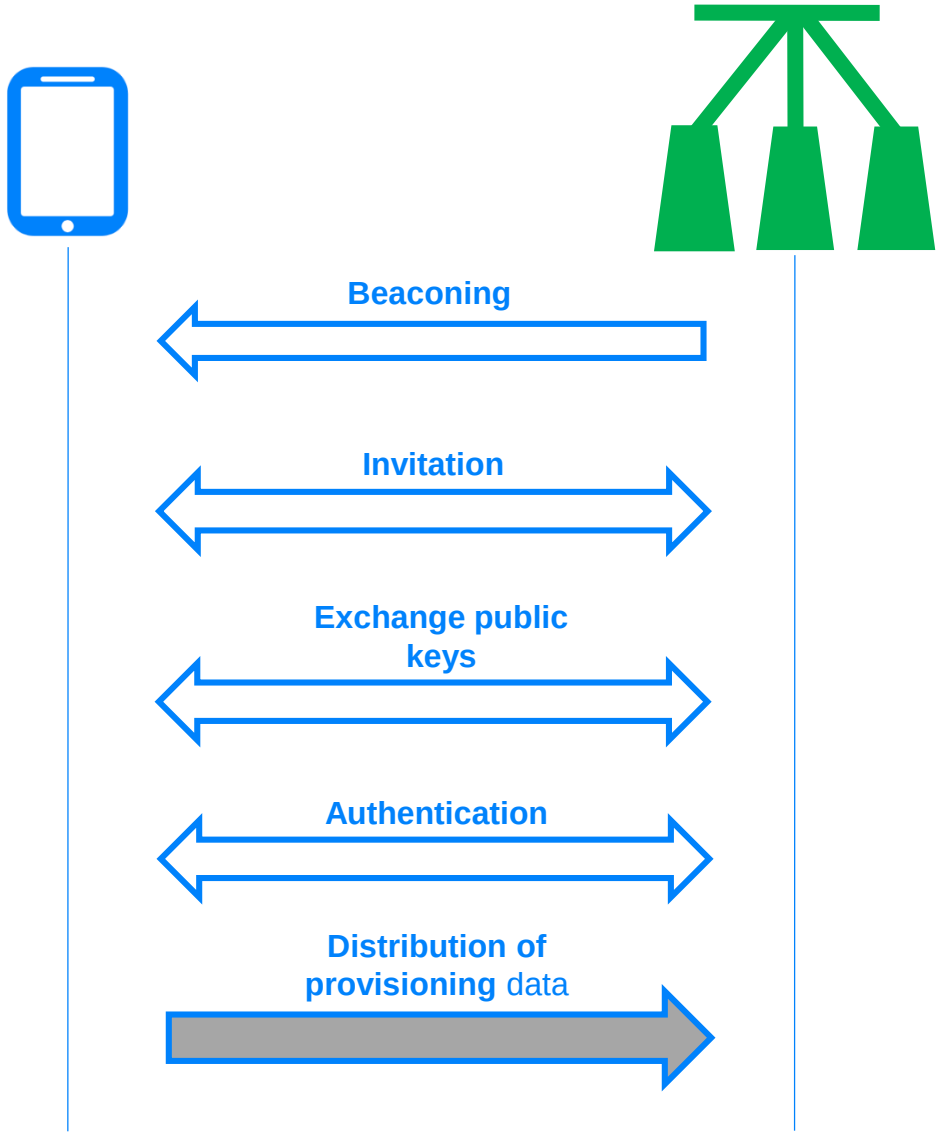
- Provisioner selects a suitable provisioning method and inform New device.
- Exchange public keys with each other.
- *ECDHSecret = P-256(private key, peer public key)*

Authentication



- 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;

Distribution

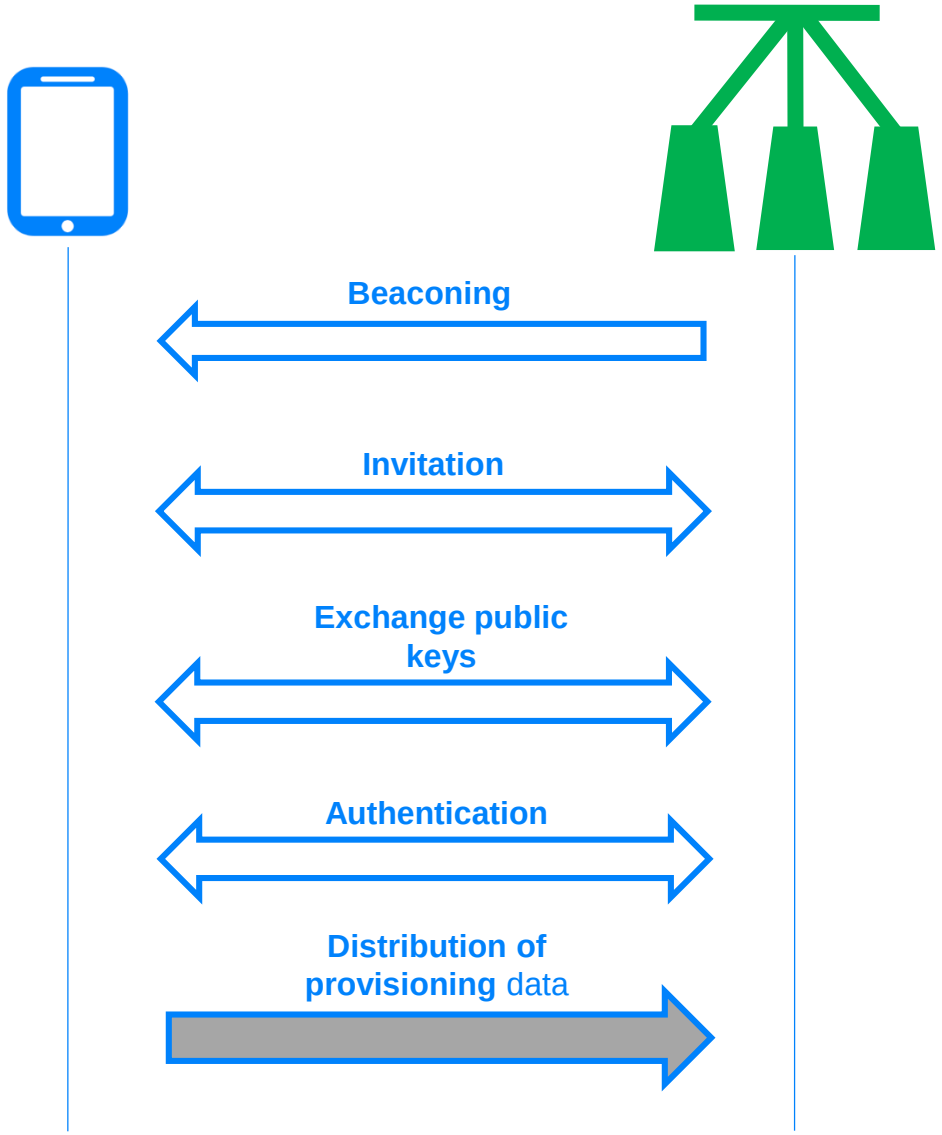


"Provisioning Data" includes:

- ❑ NetKey,
- ❑ Key index,
- ❑ Flags,
- ❑ IV Index,
- ❑ Unicast address of primary element address,

"Provisioning Data" is encrypted and distributed from Provisioner to Node.

Distribution



Section 5.4.2

https://bit.ly/2NO1V8C

"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.

Keys

NetKey

DevKey

AppKey

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

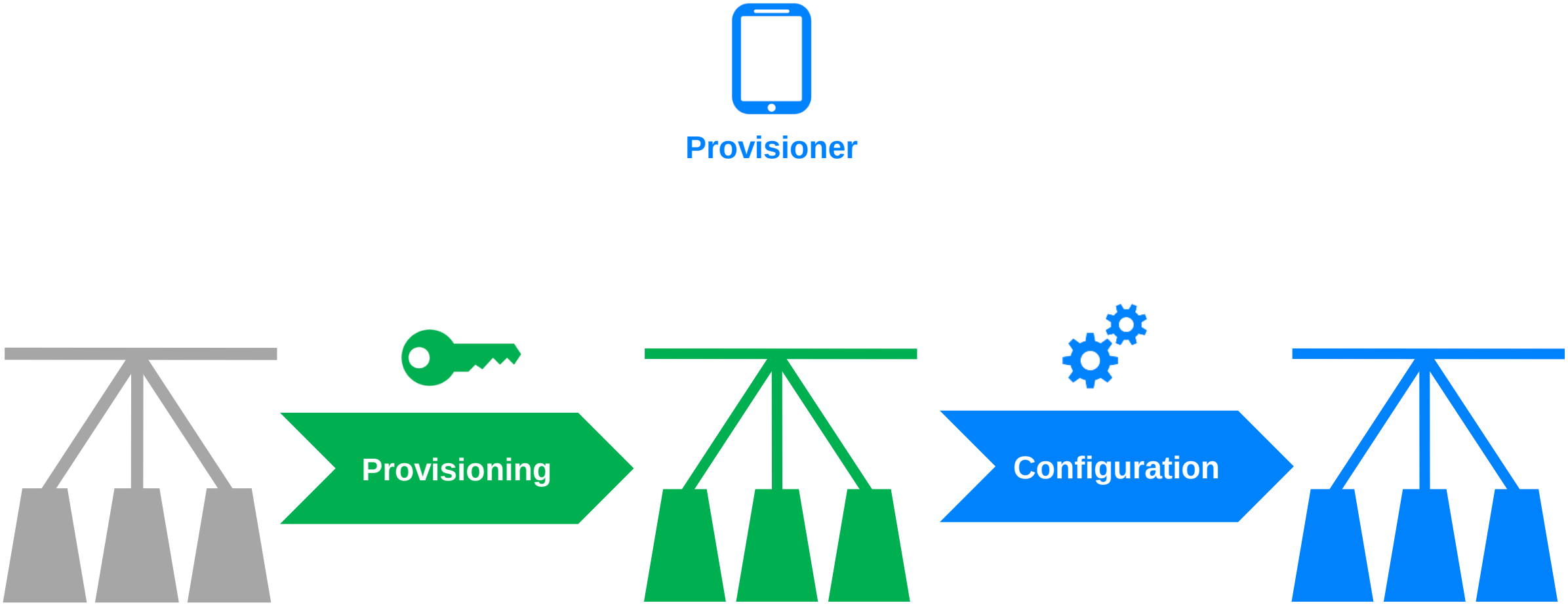
Keys

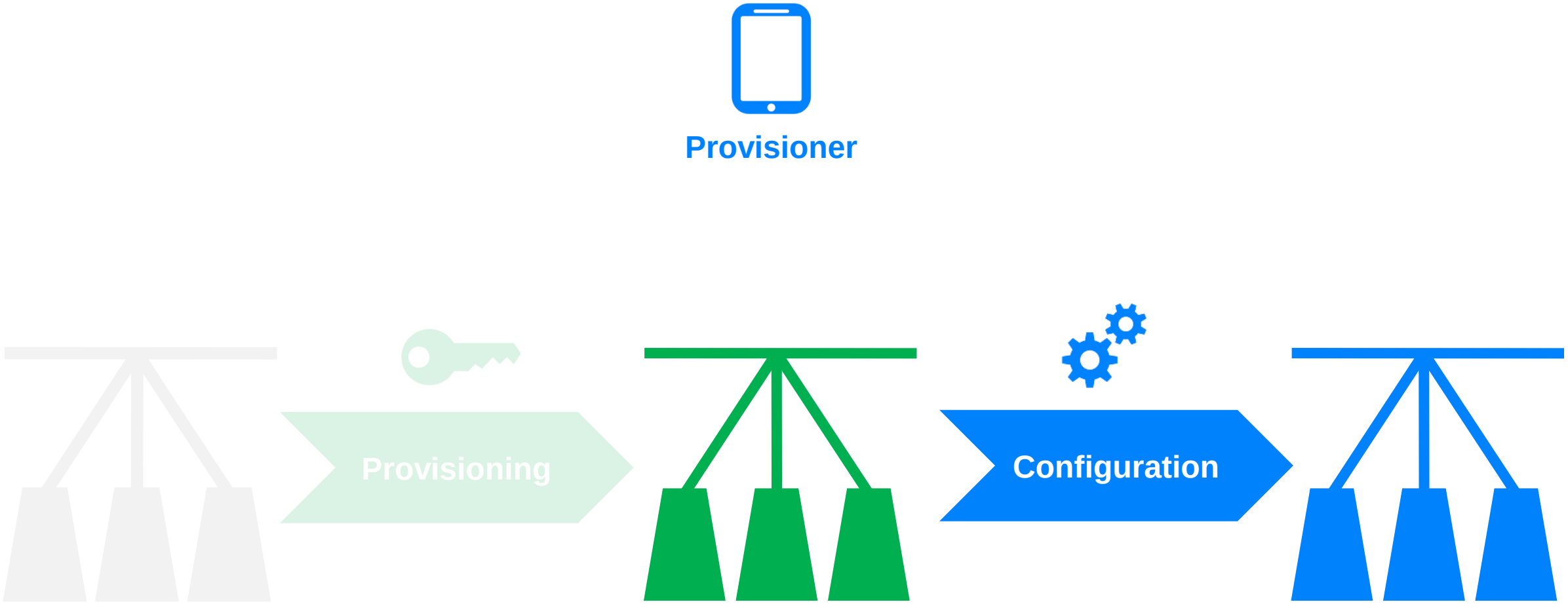
NetKey | DevKey | AppKey

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

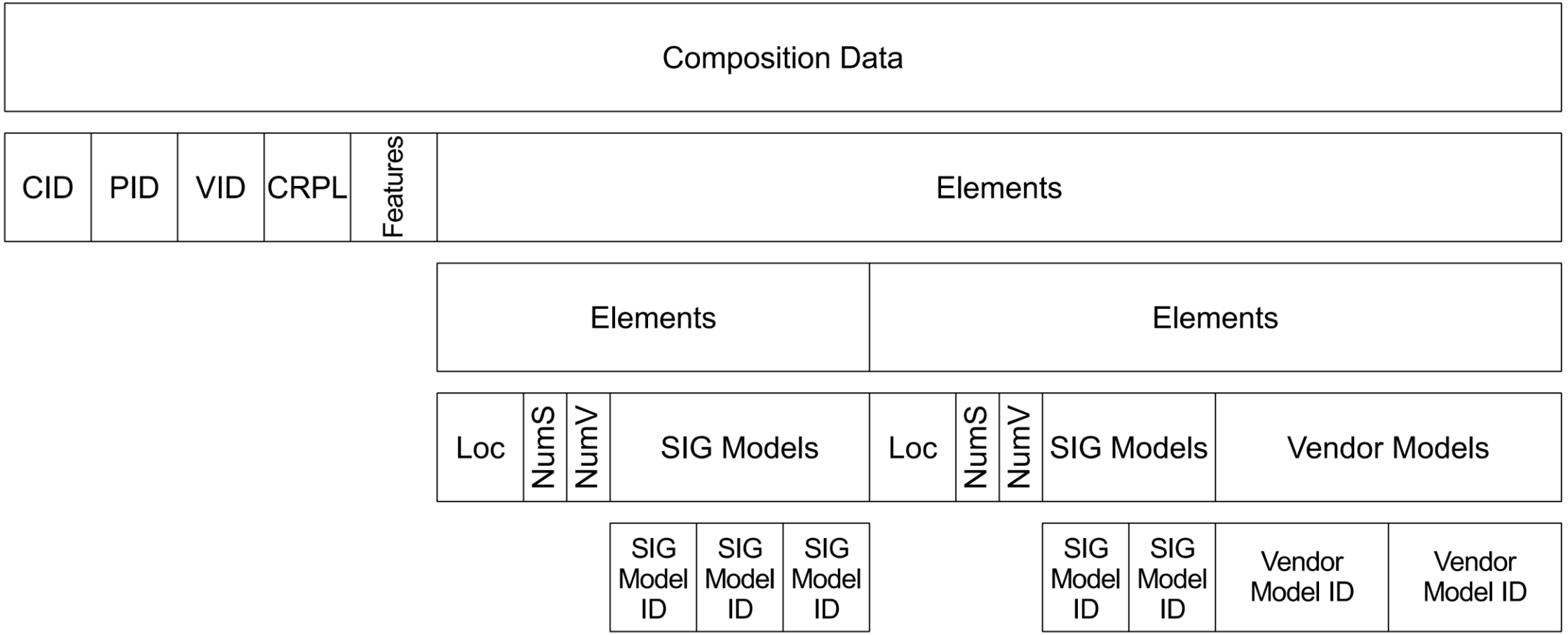
- Generate by “agreement”;
- A pair-wise key;
- Provisioner has all DevKey for each node;
- Be used to encrypt/decrypt foundation models messages;

Interoperability

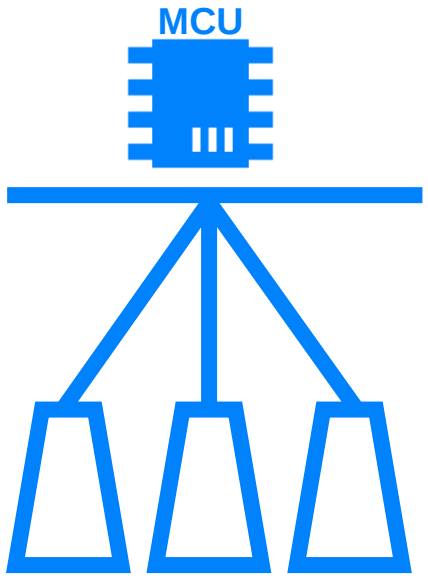

Provisioner
Provisioning
Configuration


Provisioner
Provisioning
Configuration

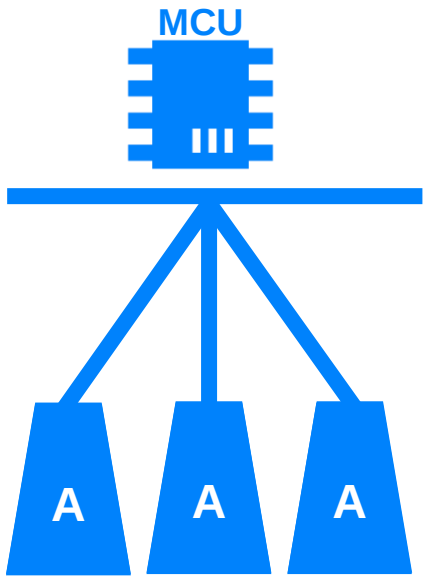
Composition Data



element

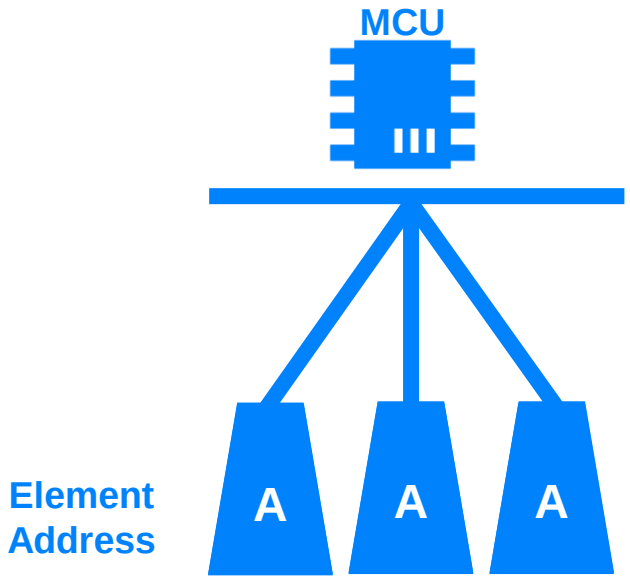


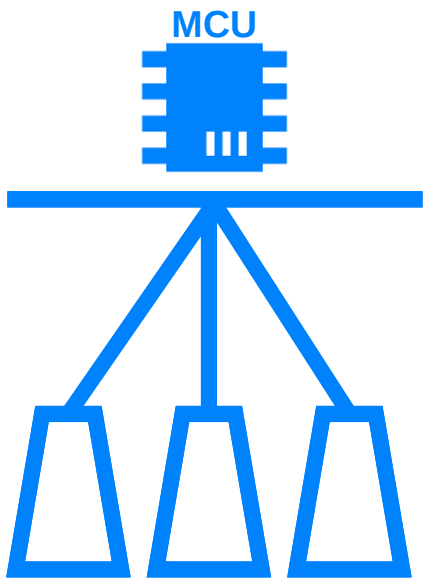
element

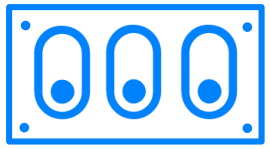


Element
Address

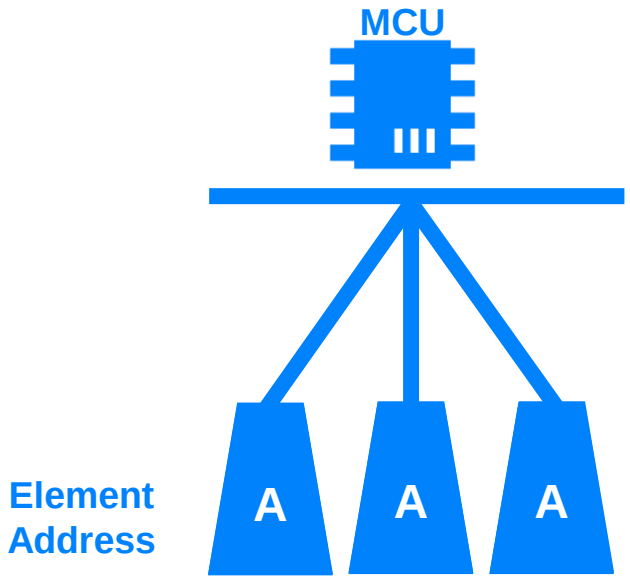
element

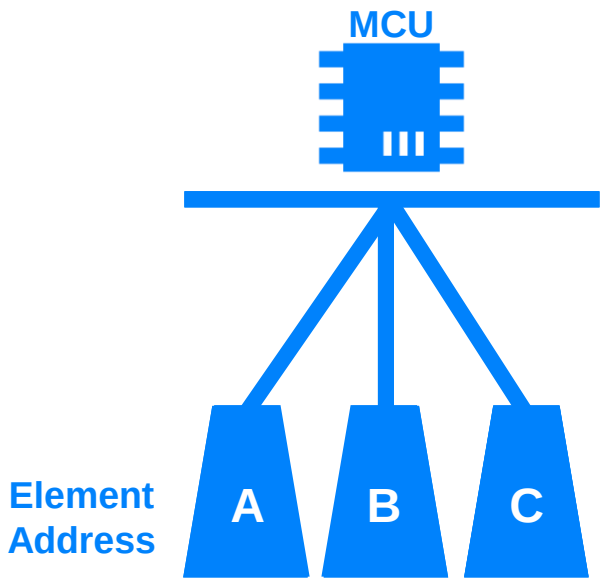


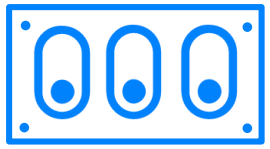




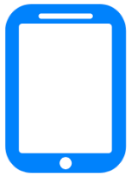
element



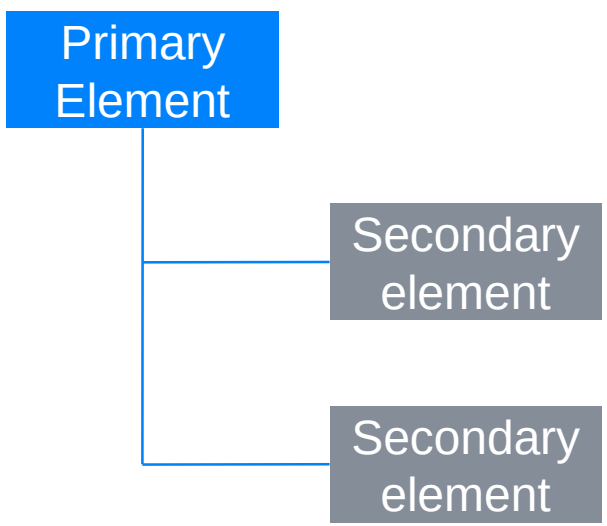


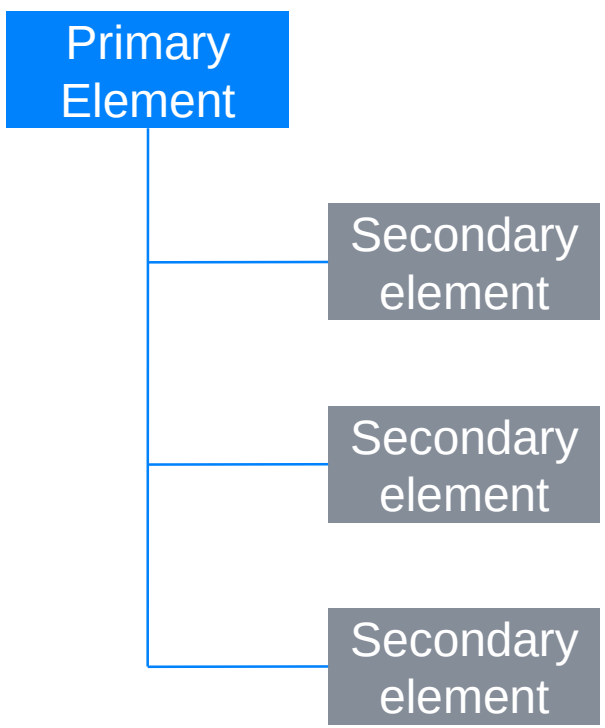


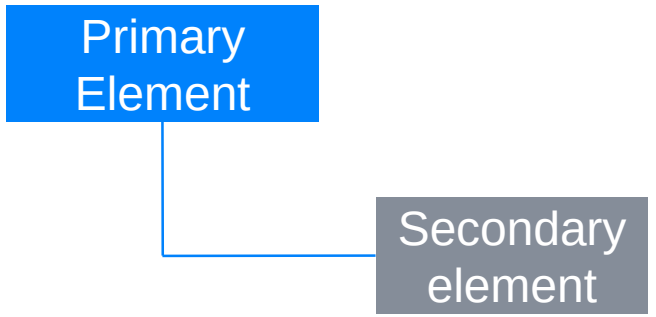
Unicast Address



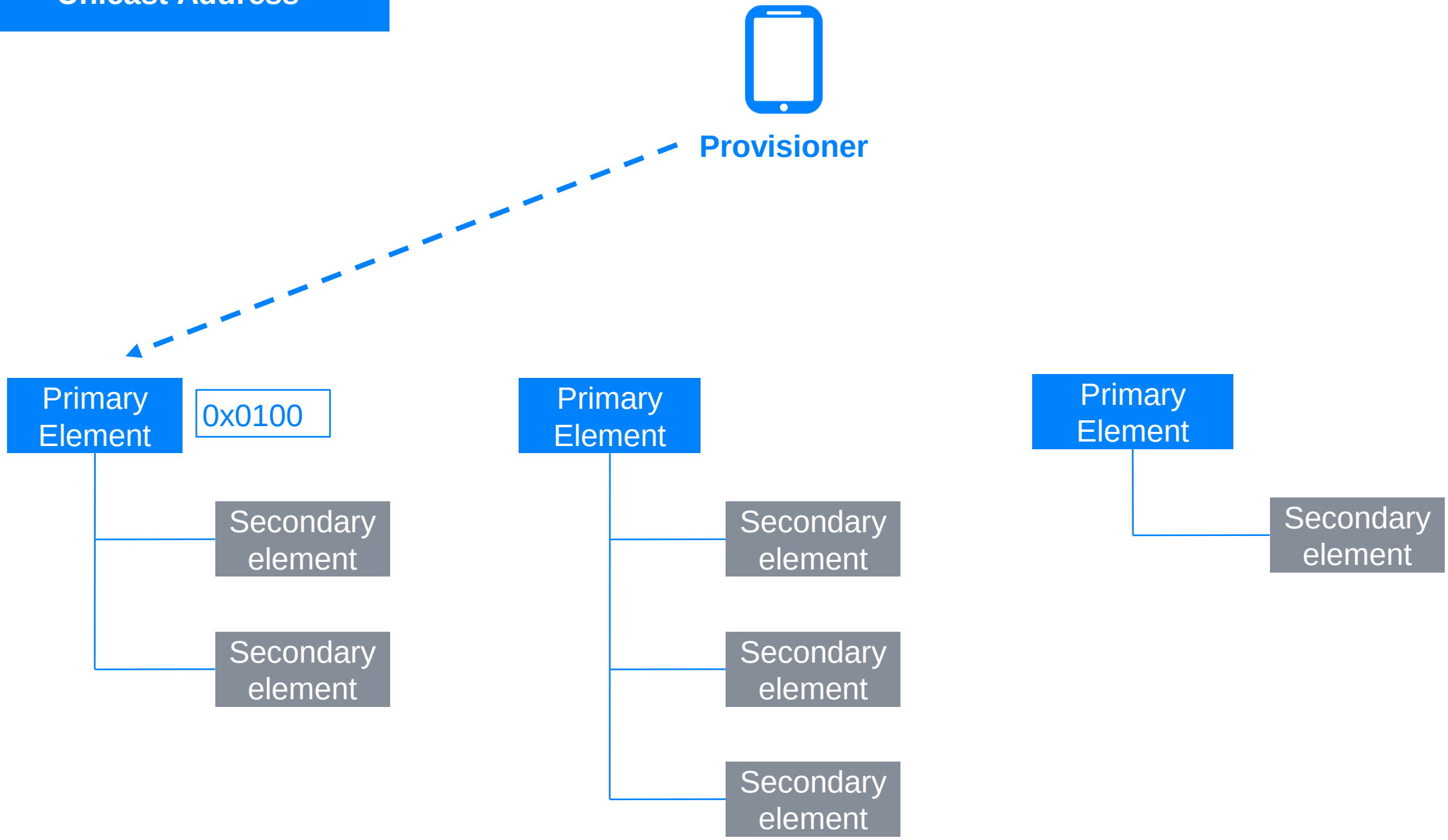
Provisioner



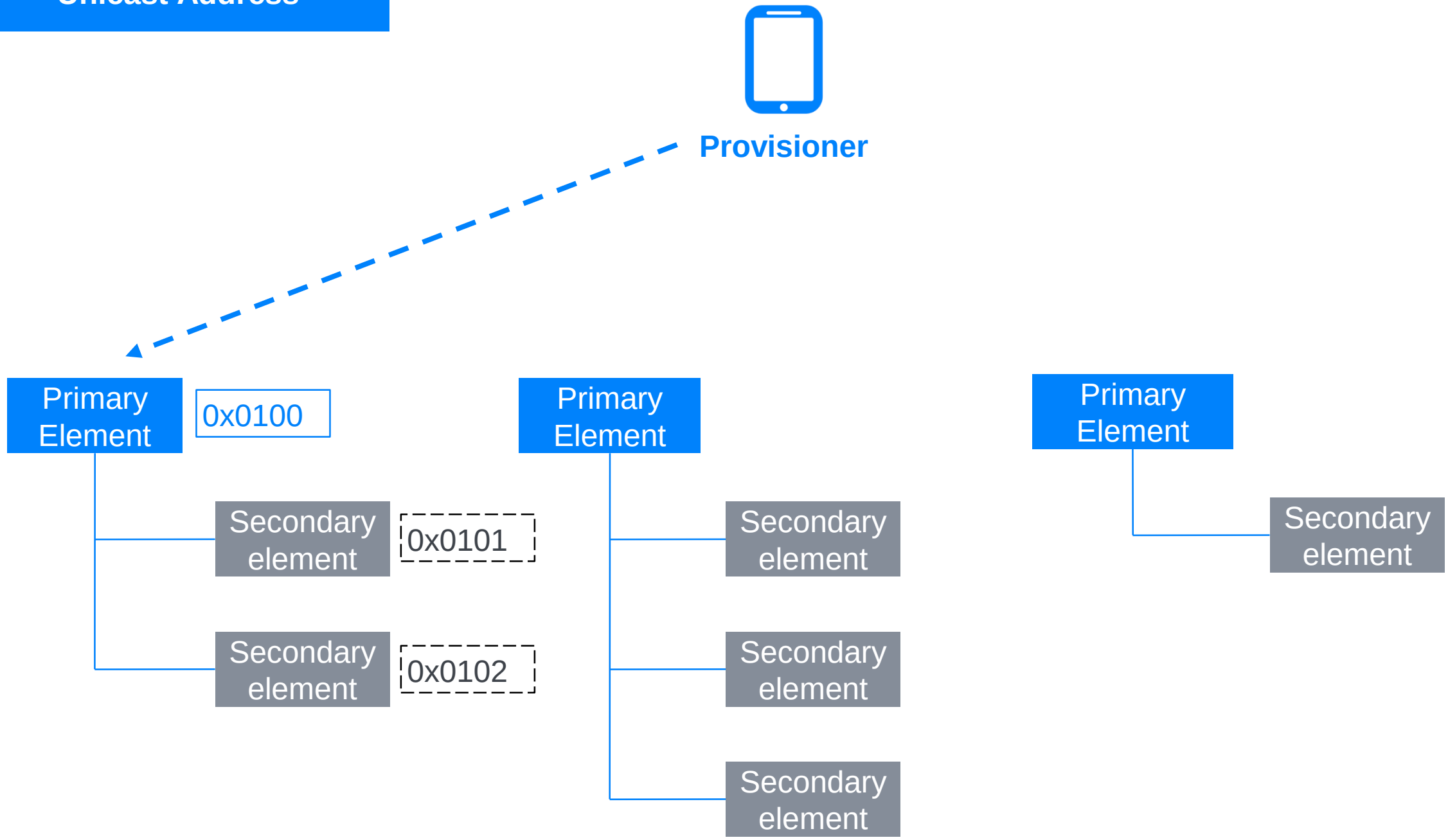




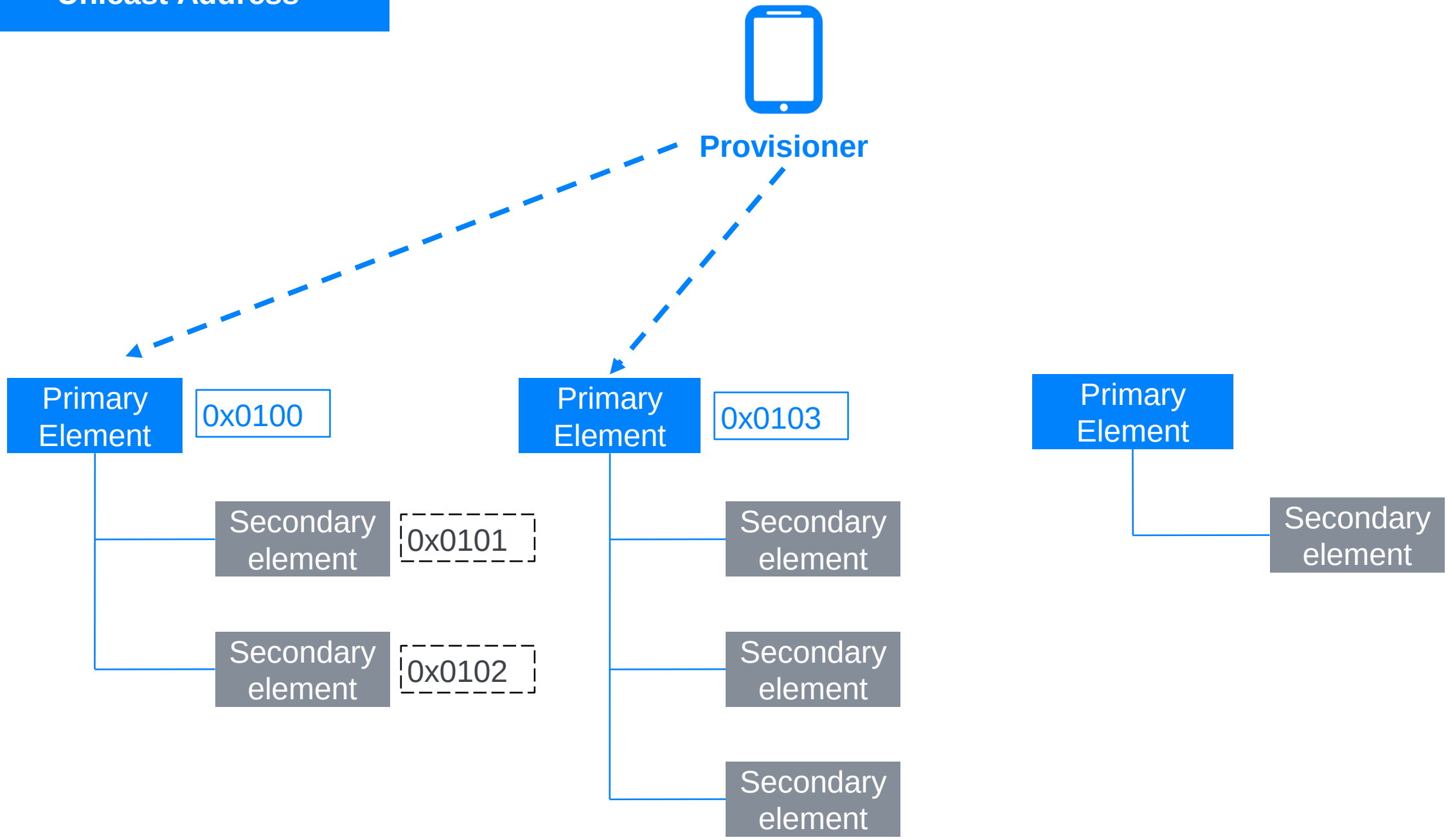
Unicast Address



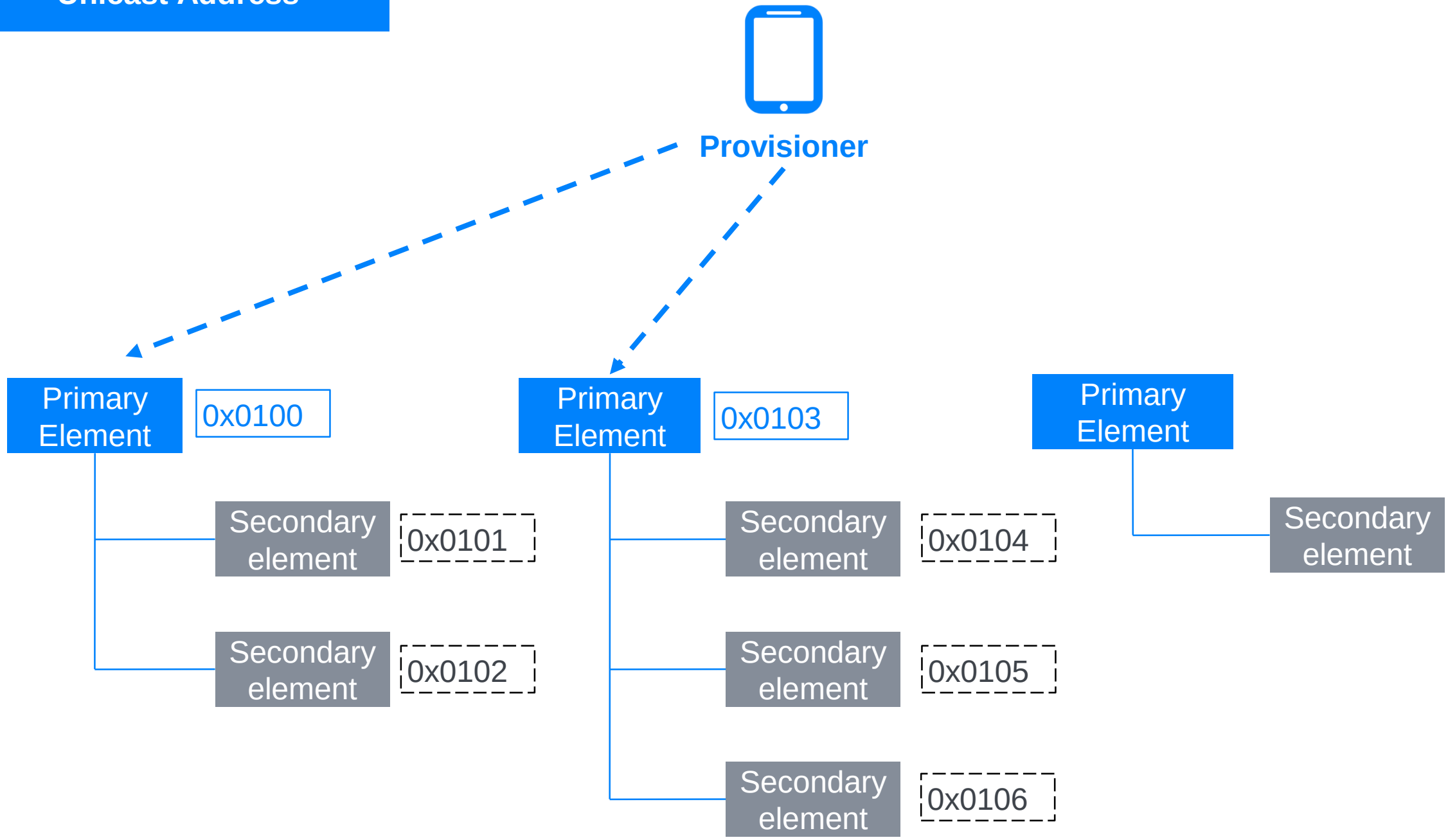
Unicast Address



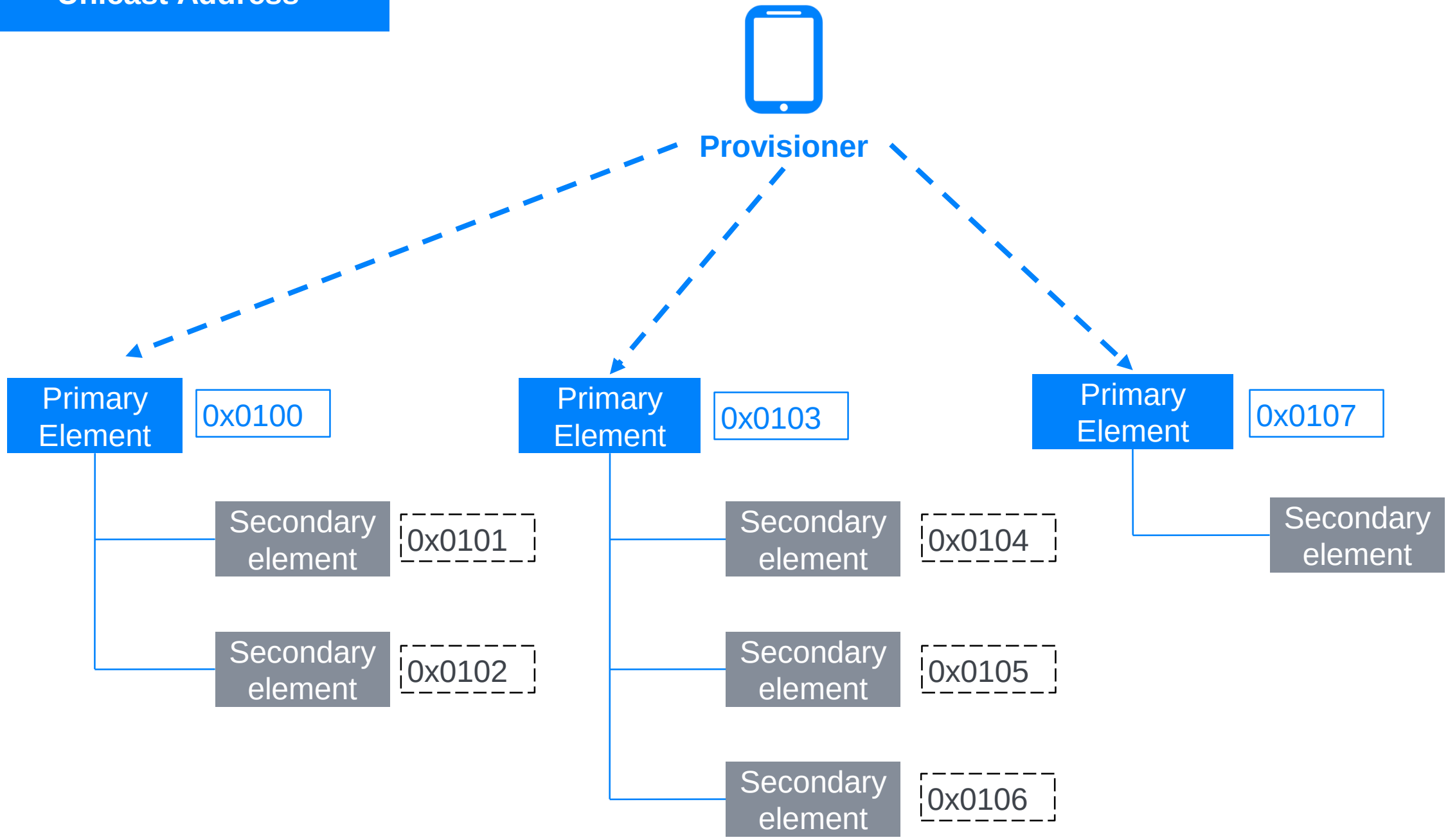
Unicast Address



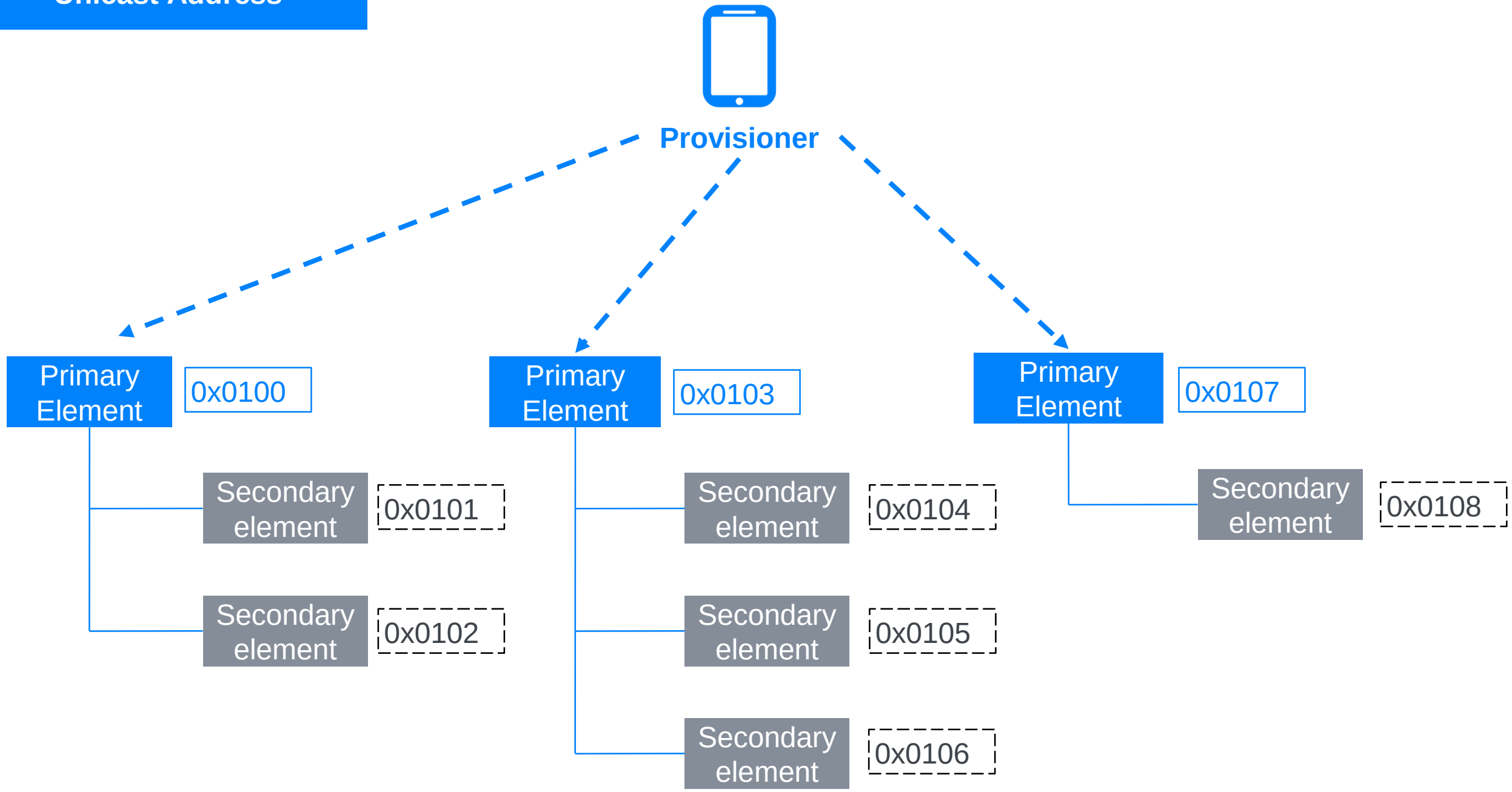
Unicast Address



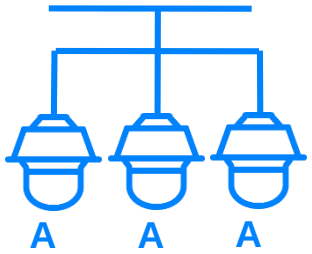
Unicast Address



Unicast Address

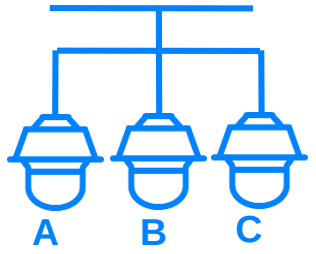


element/model



Primary Element
- Configuration Server Model
- Health Server Model
- Generic OnOff Server Model
- Vendor Model …

light



Primary Element
- Configuration Server Model
- Health Server Model
- Generic OnOff Server Model

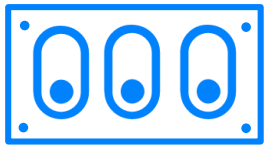
Secondary Element
- Health Server Model
- Generic OnOff Server Model
- Vendor Model …

Secondary Element
- Health Server Model
- Generic OnOff Server Model
- Vendor Model …

light

Primary element
- Configuration Server Model
- Health Server Model
- Generic OnOff Client Model
- Vendor Model …

switch



Primary Element
- Configuration Server Model
- Health Server Model
- Generic OnOff Server Model

Secondary Element
- Health Server Model
- Generic OnOff Client Model
- Vendor Model …

Secondary Element
- Health Server Model
- Generic OnOff Client Model
- Vendor Model …

switch

Keys

NetKey

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

DevKey

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

AppKey

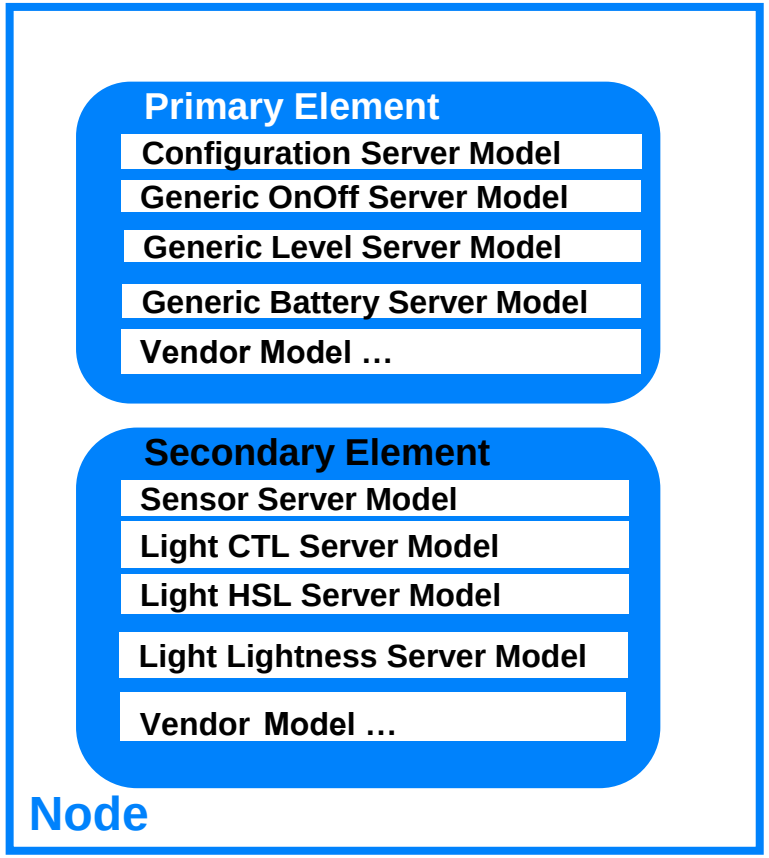
- AppKey is added by Provisioner;
- Node maintains an AppKey list;
- Up to 4096 keys in the list;
- AppKey need to bind with model at certain element;

Application Key

AppKey List

Index	AppKey
0x00	KEY0
0x01	KEY1
0x02	KEY2
0x03	KEY3
⋮	⋮
n	KEYn

AppKey & Model binding

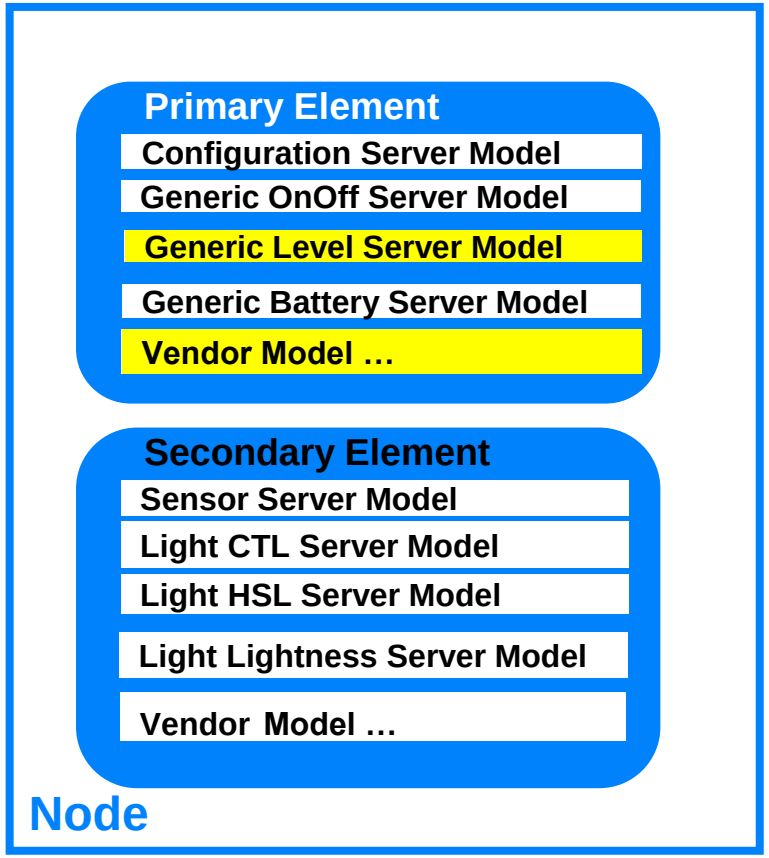


Application Key

AppKey List

Index	AppKey
0x00	KEY0
0x01	KEY1
0x02	KEY2
0x03	KEY3
⋮	⋮
n	KEYn

AppKey & Model binding

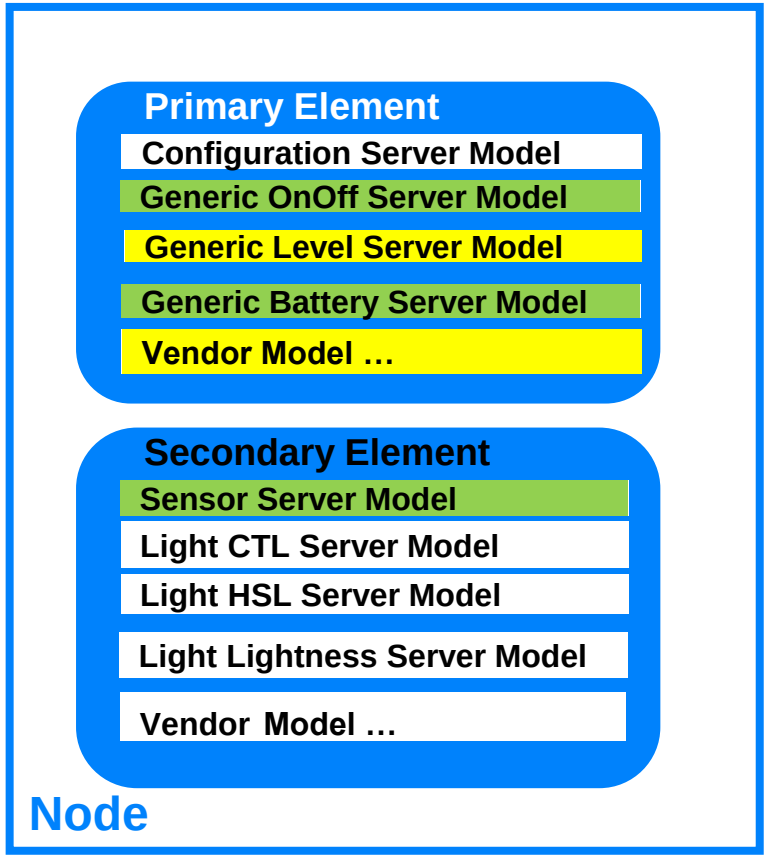


Application Key

AppKey List

Index	AppKey
0x00	KEY0
0x01	KEY1
0x02	KEY2
0x03	KEY3
⋮	⋮
n	KEYn

AppKey & Model binding

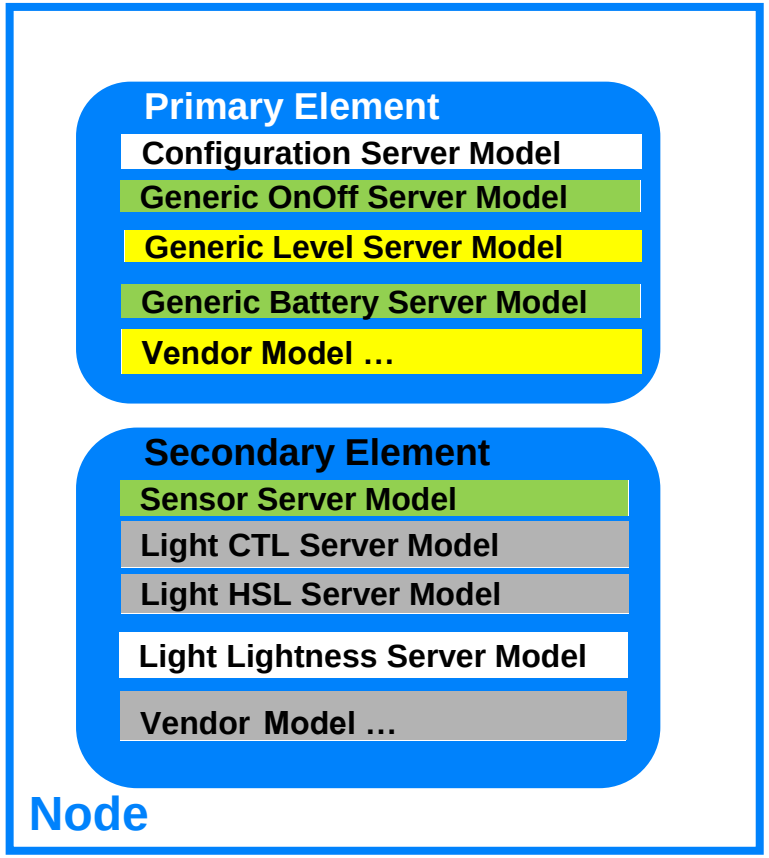


Application Key

AppKey List

Index	AppKey
0x00	KEY0
0x01	KEY1
0x02	KEY2
0x03	KEY3
⋮	⋮
n	KEYn

AppKey & Model binding

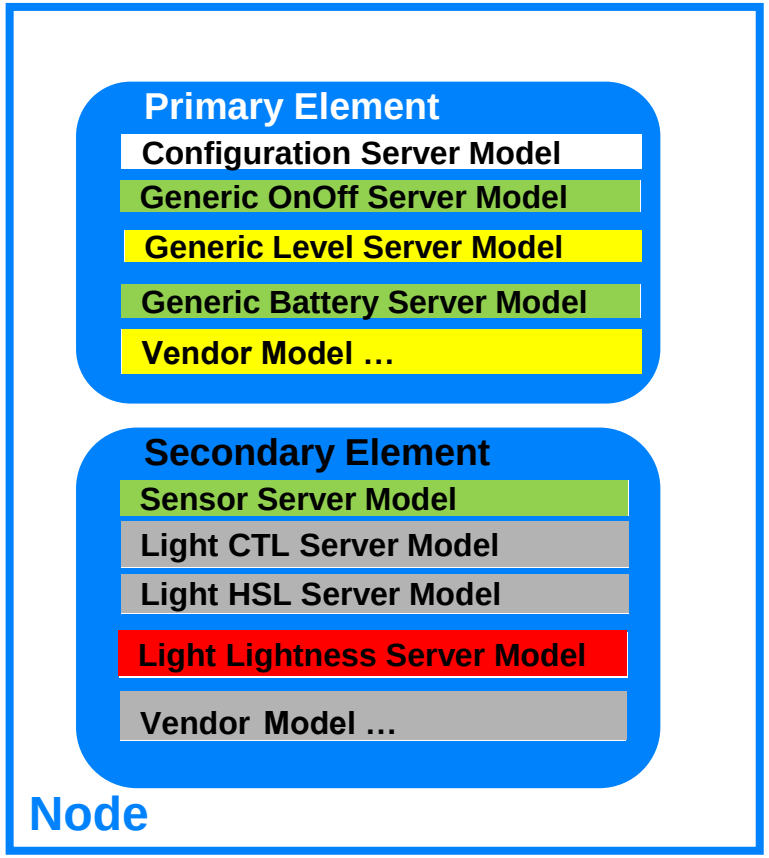


Application Key

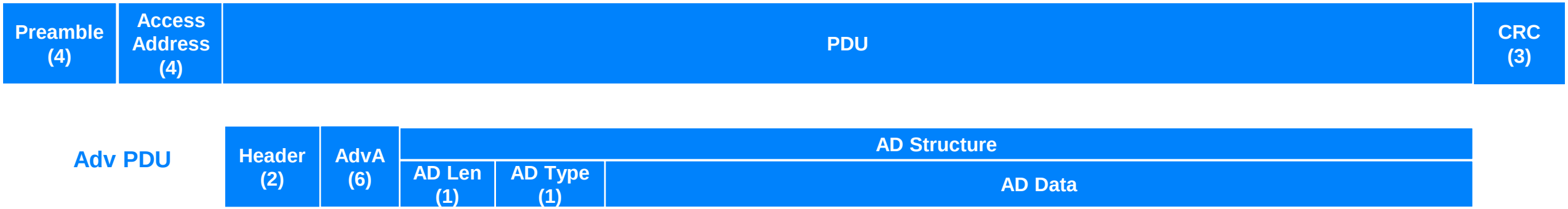
AppKey List

Index	AppKey
0x00	KEY0
0x01	KEY1
0x02	KEY2
0x03	KEY3
⋮	⋮
n	KEYn

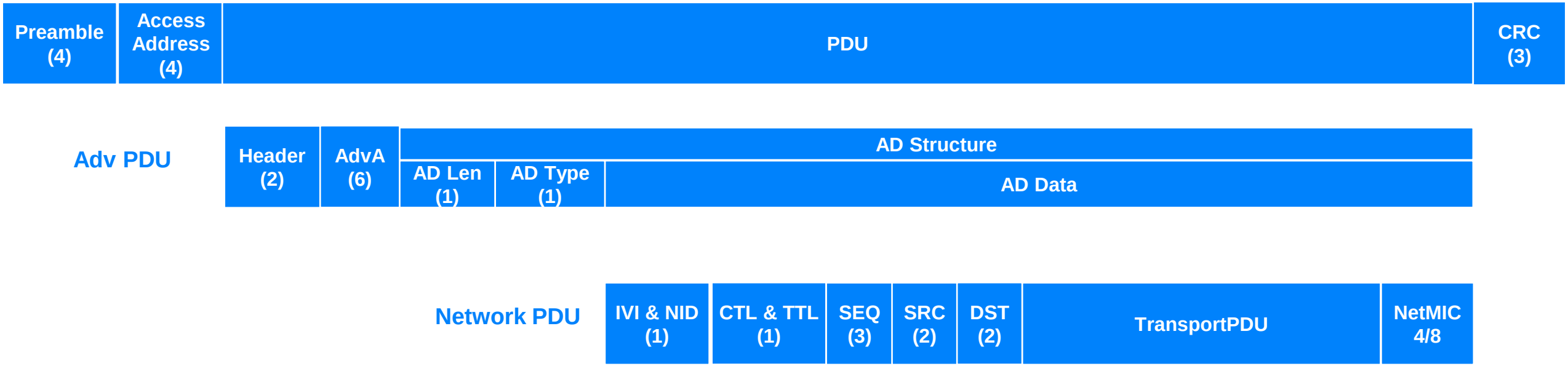
AppKey & Model binding



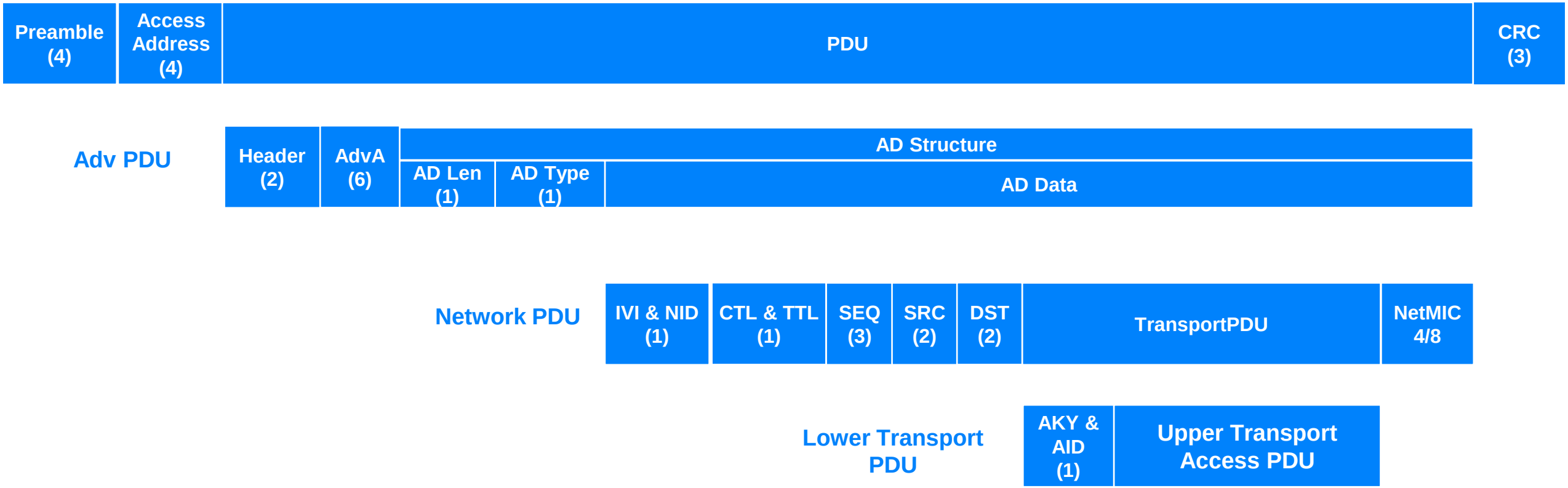
Unsegmented access message



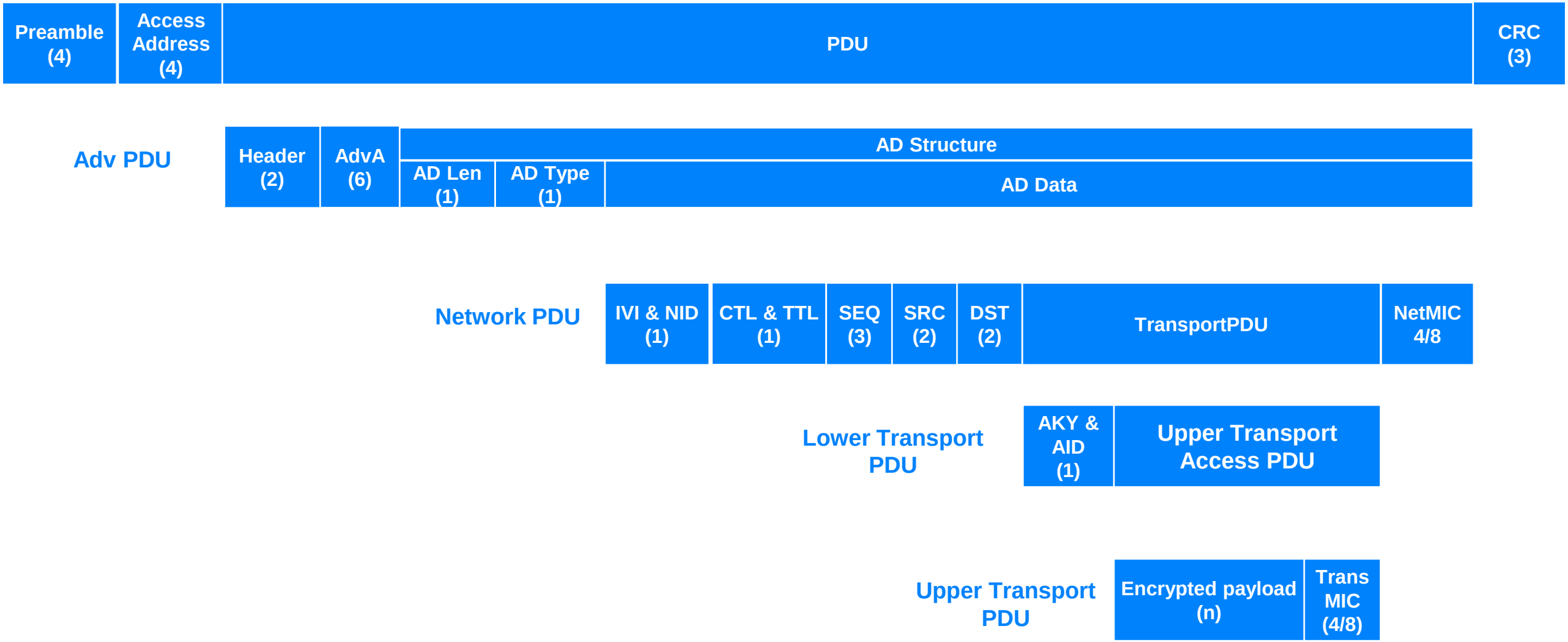
Unsegmented access message



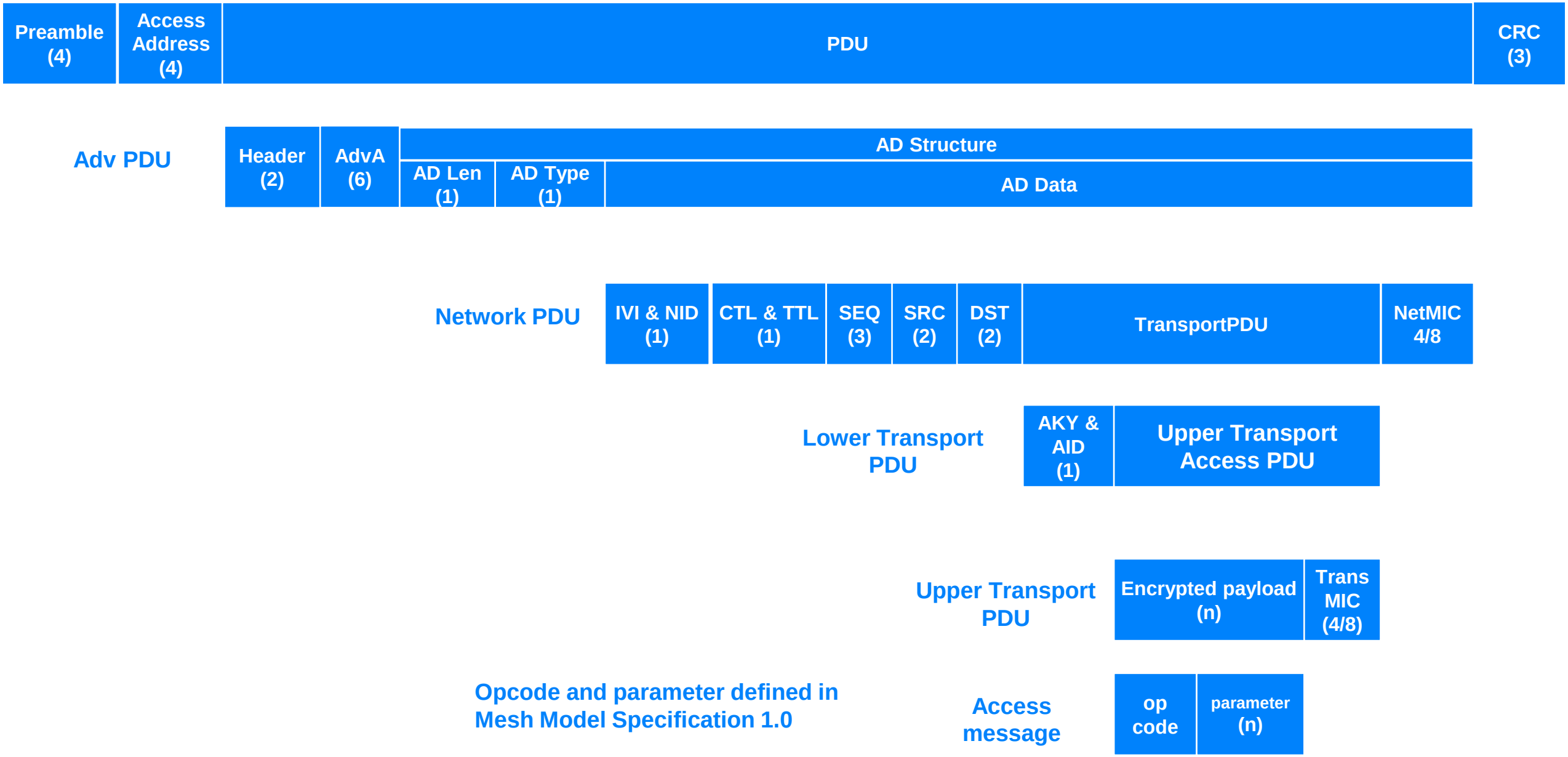
Unsegmented access message



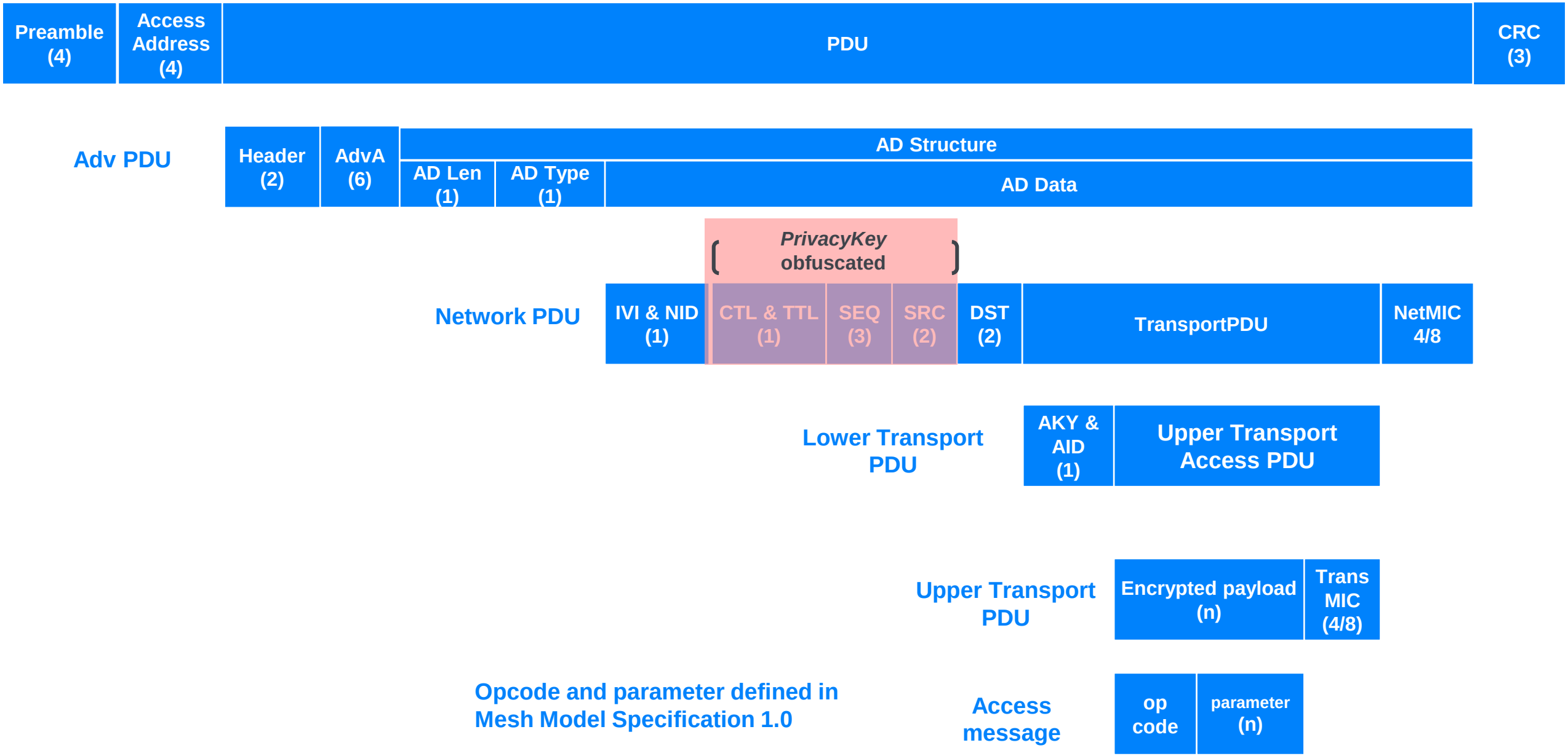
Unsegmented access message



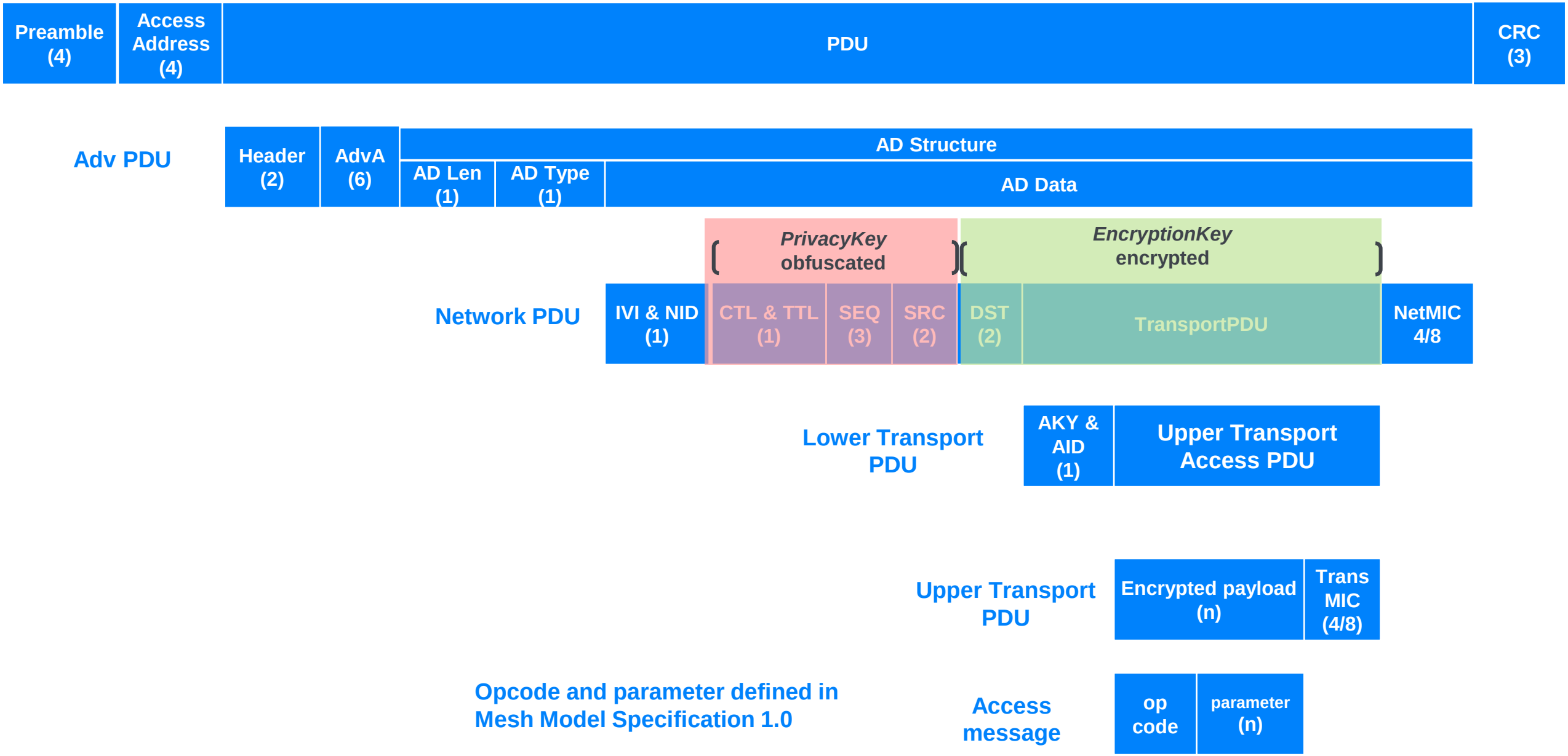
Unsegmented access message



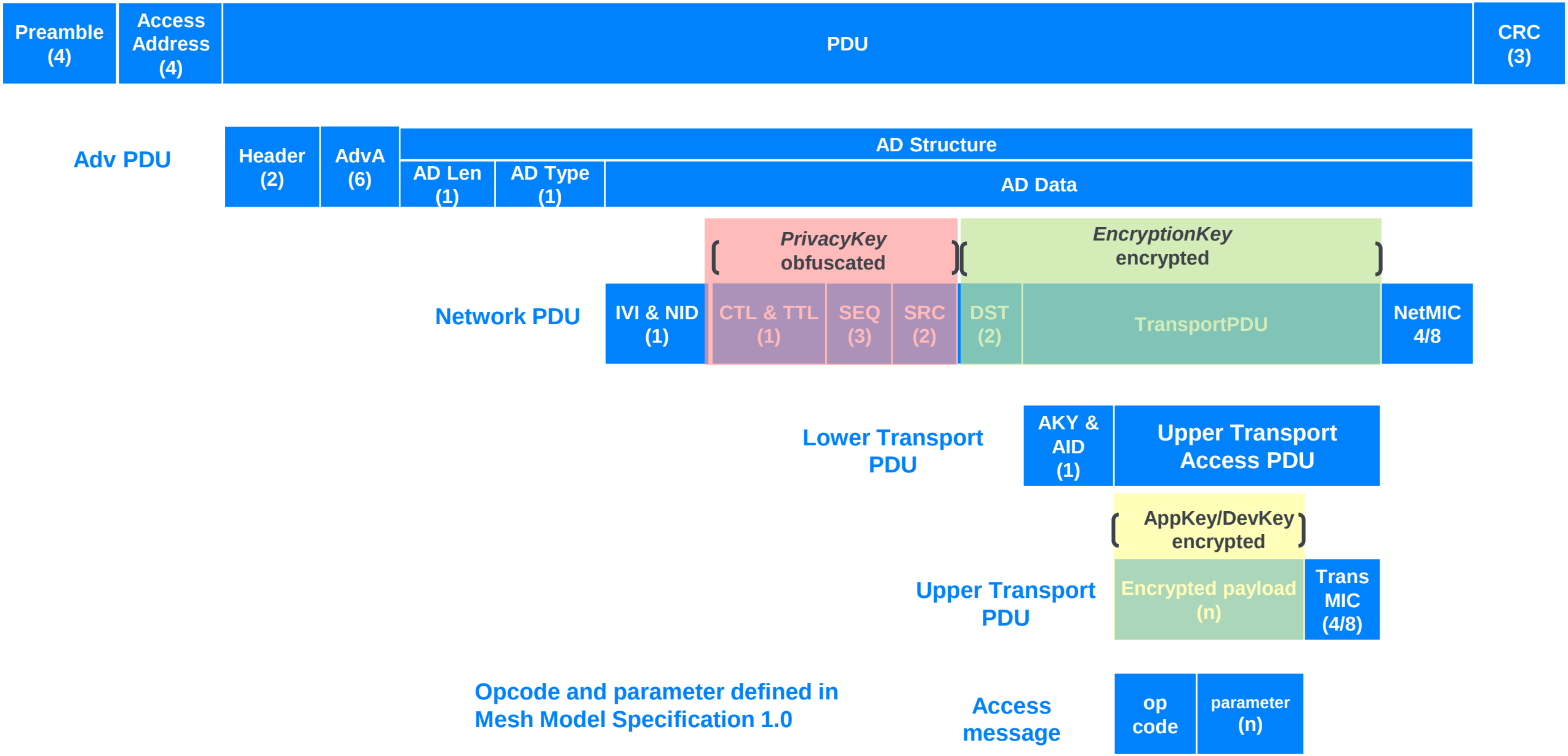
Unsegmented access message



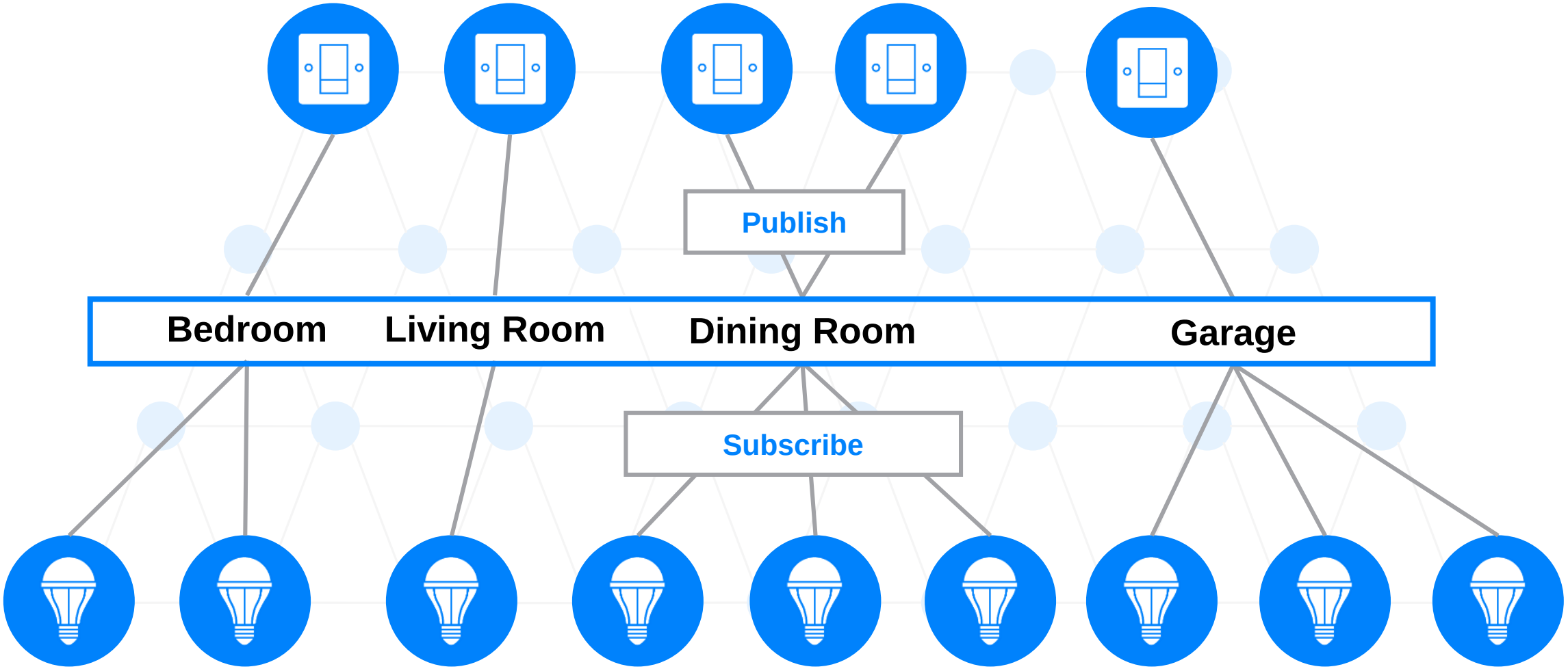
Unsegmented access message



Unsegmented access message



Publish/Subscribe



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;*

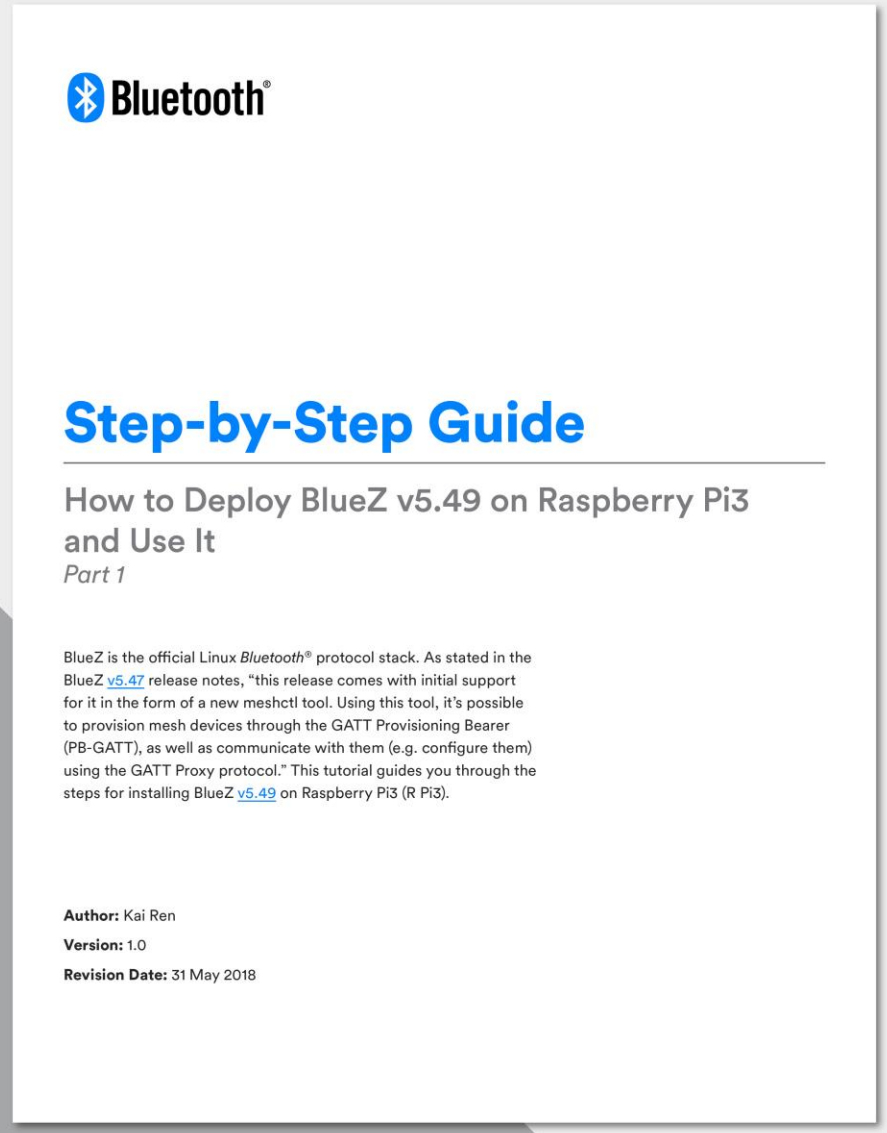
Demo

Interoperability Demo

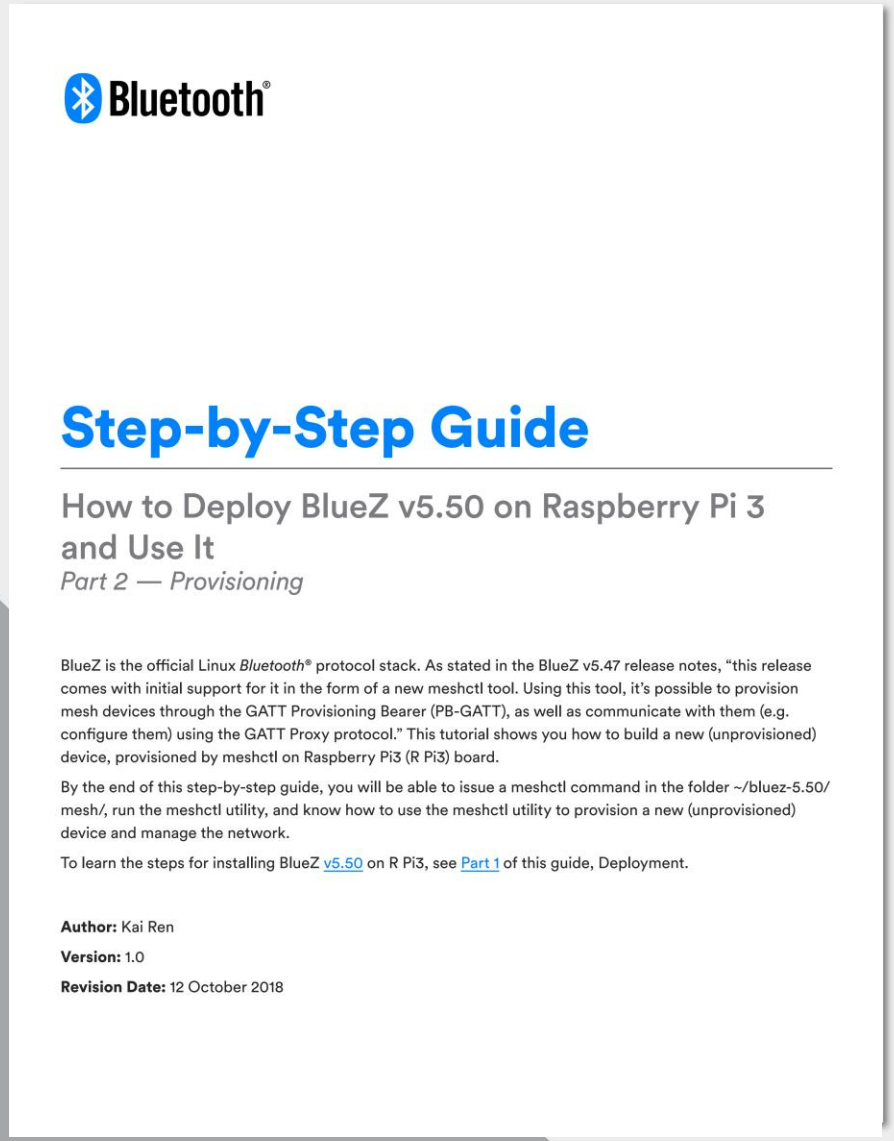
Provisioner	Kit	Stack
Apple Watch 3	**Nordic** nRF52840 Dev Kit, PCA10056	Zephyr v1.14.0
iPhone 8	**Silicon Labs** EFR32 Blue Gecko Bluetooth Starter Kit	Bluetooth mesh v1.2.0
	Cypress CYW920719Q40EVB-01	WICED 6.20
Pixel 2	**STMicroelectronics** STEVAL-IDB008V2	BlueNRG-Mesh V1.06.00
	Micro:bit Education Foundation micro:bit board	Zephyr v1.14.0

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

Now available at https://bit.ly/2MMgoFN



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



Bluetooth®

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

Now available at
https://bit.ly/2K5UeM7

#BluetoothAsia2019#

BluetoothAsia
探索、创新、开拓

谢谢

Thank you!