

Chapter 7A: Bluetooth Mesh Topology and Client Applications

This chapter covers the basics of the Bluetooth mesh network topology.

7A.1 OVERVIEW	2
7A.2 MESH SPECS	2
7A.3 NODES	3
7A.3.1 STANDARD NODE	4
7A.3.2 RELAY NODE	4
7A.3.3 GATT PROXY NODE	5
7A.3.4 FRIEND AND LOW POWER NODES	5
7A.4 PROVISIONING AND CONFIGURATION/MANAGEMENT	6
7A.4.1 PROVISIONING	6
7A.4.2 CONFIGURATION/MANAGEMENT	8
7A.5 CYPRESS MESH HELPER PROGRAMS	9
7A.5.1 CLIENT CONTROL MESH HOST APP (WINDOWS ONLY)	9
7A.5.2 MESH CLIENT HOST APP (WINDOWS, MACOS, LINUX)	13
7A.5.3 MESH CLIENT PEER APP (WINDOWS 10 ONLY)	14
7A.5.4 CYPRESS MESH CONTROLLER APP (ANDROID)	16
7A.5.5 CYPRESS MESH APP (IOS)	18
7A.6 DEMO	20
7A.7 EXERCISES	20
EXERCISE 7A.1 CREATE NETWORK WITH A DIMMABLE LIGHT MESH DEVICE	20

7A.1 Overview

Traditional Bluetooth LE devices use point-to-point communication. That is, each pair of devices send data back and forth to each other. Each of these connections has a GAP Central and a GAP Peripheral.

In contrast, in a mesh network every device in the mesh can communicate (either directly or indirectly) with every other device in the network. Some devices in the network can relay messages that they receive so that the overall communication range is extended beyond the radio range of each individual device. In theory, the range of a mesh network is unlimited as long as you have at least one relay device within range of every device in the network.

In Bluetooth Mesh, messages are sent using advertising packets. That is, no connections are made. Rather, data is broadcast by a sending device using advertising packets which can be received by any devices that are in range of the sender.

Devices in a mesh network are called "nodes". Devices that are not part of a mesh network (yet) are called "unprovisioned" devices. The process of provisioning a node will also be covered later.

A mesh network can have one or more subnets that enable isolation of related groups of nodes. A subnet is a group of nodes that can communicate with each other at the network layer because they share a network key. The difference between a network and a subnet is that a node may belong to more than one subnet by having more than one network key.

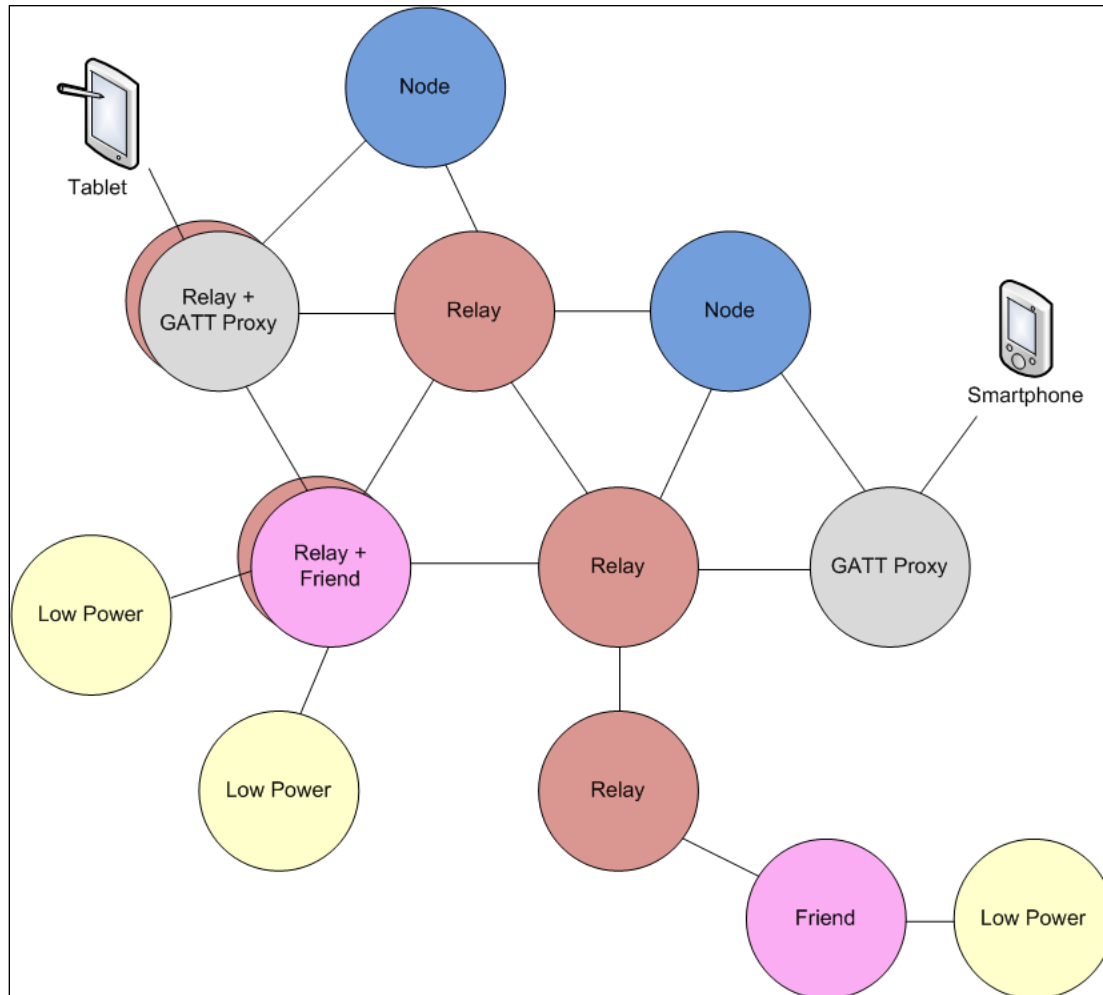
7A.2 Mesh Specs

Before going into more detail, it is worth noting that the Bluetooth SIG provides three specifications that contain every detail of the mesh protocol. These are:

- [Mesh Profile](#) – defines fundamental requirements for mesh networking
- [Mesh Model Specification](#) – defines models which are used to define basic functionality of nodes in a mesh network
- [Mesh Device Properties](#) – defines device properties required for the mesh model spec

7A.3 Nodes

The following figure shows an example mesh network topology. Each of the types of node will be discussed in detail in the following sections. It is suggested that you refer to this figure while reading the descriptions.



Each node in a mesh network can send and receive messages. Each node may also implement one or more of the following features depending on its capabilities:

- Relay
- GATT Proxy
- Friend
- Low Power

Relay, GATT Proxy and Friend features can all be implemented on the same node. Typically, it doesn't make sense for a Low Power node to implement any of the other features as you will see in a minute.

7A.3.1 Standard Node

The standard node functionality involves sending and receiving mesh messages. Every node in the network must be able to act as a standard node.

Message Caching

Each node must maintain a message cache containing all recently received messages. If a message is received more than once, it is immediately discarded. In this way, if a message is relayed by multiple nodes to a final destination, the destination only acts on the message one time.

7A.3.2 Relay Node

Relay nodes can receive a message for the network and then retransmit it to other devices in range. This is the method by which mesh networks can cover larger distances than the range of any single device. For a network to operate, every node must be within range of at least one relay so that its messages can be forwarded on to nodes that it cannot directly communicate with.

It is common for all except low power nodes to implement a relay feature in order to maximize the possible paths through a mesh network.

Due to the message caching described above, a relay node will only relay a given message one time.

TTL

Each message has a field called the Time To Live (TTL). This is used to determine how many times a given message will be retransmitted. By understanding the basic topology of a mesh network, the TTL can be used to prevent messages from being retransmitted too many times. This allows the mesh network to be more efficient.

In fact, there are heartbeat messages sent periodically which include, among other things, information that allow receiving nodes to determine how many hops away the sender is. Networks can use this information to adapt TTL settings to optimize the network.

Security

A relay node only decodes enough of the message to decide what to do with it. For example, it decodes the addresses for the message but not the payload if it is not intended for that node. In fact, due to the security architecture, the relay node cannot decode the payload for any messages that are not from the same network application (e.g. lighting). Security will be discussed in detail later.

7A.3.3 GATT Proxy Node

Many existing BLE devices support traditional BLE GATT communication but not mesh communication. Most smartphones and tablets fall into this category. Since you may want to interact with a mesh network from one of those devices, the GATT proxy was created. A GATT proxy node has both a mesh interface and a GATT interface. The GATT interface is used to communicate with BLE devices that don't possess a mesh stack and then relay those messages to/from the mesh network. That is, the GATT proxy acts as a bridge between the mesh network and the traditional BLE GATT device.

7A.3.4 Friend and Low Power Nodes

Friend and Low Power Nodes are used to optimize power consumption for constrained devices. Devices that are power constrained (e.g. a battery powered device) are designated as low power nodes. Every low power node in the network must be associated with exactly one friend node. Friend nodes are devices which are not power constrained (e.g. a device plugged into AC power) that support 1 or more low power nodes depending on its capabilities (e.g. available RAM).

When a low power node is added to a mesh network it broadcasts a request for a friend. Each friend in range that can handle a new low power node replies and the low power node selects the best friend based on how many messages the friend can store; the RSSI and the timing accuracy.

Once the relationship is established, the friend node will receive and store messages for any low power nodes that it is associated with. The low power node will periodically ask the friend node for any messages that the friend has stored for it. In this way, the low power node does not need to listen continuously for mesh packets. Instead, it can be in a low power mode most of the time and can wake up only periodically for a very short time.

For example, consider a battery powered mesh connected thermostat. It will measure the actual temperature and may send a mesh message with the temperature once per minute. This can be done with very low power consumption since the device can be sleeping all the time except for a short period each minute to send the value. However, it must also be possible to change the set point of the thermostat. In this case, instead of sending messages, the thermostat must be listening for messages. If it listens constantly for messages the power consumption will be unacceptably high, but if it only listens occasionally for messages it will likely miss messages. By making the thermostat a low power node we get the best of both worlds - it can send messages once a minute and receive any stored messages regarding the set point from its friend node. No messages are missed even though the thermostat is awake only a very small percentage of the time.

7A.4 Provisioning and Configuration/Management

To get a new device up and running on a Bluetooth Mesh network, it must be provisioned and configured. These are often thought of as a single step, but they are unique processes with different protocols. These will each be described separately below.

7A.4.1 Provisioning

Provisioning is the process by which a device is made a member of the mesh network and becomes a node. To be a node on a mesh network, a device needs to have the network key (and other associated network security information like the IV index) and it needs to have a unicast address assigned to its primary element. Provisioning can be done using either a GATT connection (PB-GATT) or an advertising channel (PB-ADV) as the bearer. (PB = Provisioning Bearer).

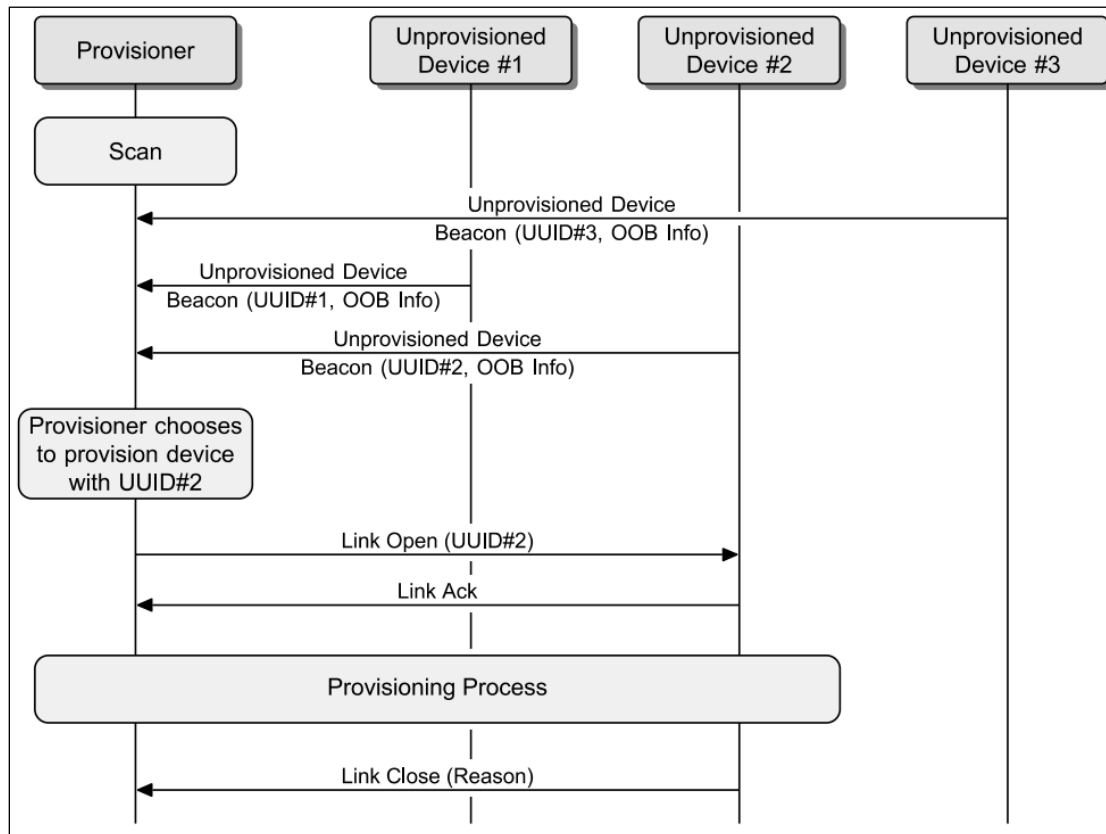
Provisioning is most commonly done using an application on a smartphone or a tablet. Note that smartphones currently do not support provisioning over an advertising channel, so from a practical standpoint, all devices should support provisioning over GATT. The Bluetooth Mesh spec strongly recommends that unprovisioned devices support both.

Beaconing

Any unprovisioned device will indicate its availability to be provisioned by sending out advertising packets of the type "Mesh Beacon".

Link Establishment

A provisioner will scan for unprovisioned devices and will choose (usually via input from the user) which device to provision. The provisioner sends a Link Open message to the device to be provisioned which will in turn respond with a Link ACK message. Once provisioning completes, the provisioner sends a Link Close message. These steps are illustrated in the figure below.



(This figure is taken from the Bluetooth Mesh Profile Specification)

The remaining steps detailed below occur within the box labeled "Provisioning Process" in the figure above.

Invitation

The provisioner sends an invitation to the device being provisioned in the form of a provisioning invite protocol data unit (PDU). The device being provisioned responds with information about itself in the form of a provisioning capabilities PDU.

Exchanging Public Keys

The provisioner and the device to be provisioned exchange public keys either directly or using an out-of-band (OOB) method.

Authentication

Authentication is performed using an OOB method that depends on the capabilities of the device being provisioned. For example, if the device to be provisioned has some output mechanism, it creates a random number and indicates that number to the user (e.g. it may flash an LED a random number of times, beep a random number of times, or show the number on a display). The user then enters that number into the provisioner.

If the device has some input mechanism, then the provisioner creates a random number and presents it to the user. The user then inputs that number on the device (e.g. by pressing a button the specified number of times or entering the number using a keypad).

Either way, once the random number has been generated on one side and entered on the other, a cryptographic exchange happens between the two devices using that random number.

Distribution of Provisioning Data

Once authentication is done, a session key is derived by each device from its private key and the public key from the other device. The session key is used to secure subsequent distribution of the data needed to complete provisioning. Once provisioning is completed, the provisioned device has the network's key (NetKey), a security parameter called the IV index, and its Unicast address which was allocated by the provisioner. The device is now a node and is a part of the network. The provisioner then sends a Link Close message as described previously.

7A.4.2 Configuration/Management

Once provisioning is done, the same smartphone or tablet (i.e. the provisioner) then uses the mesh network to configure the new node. This includes distribution of application keys, assigning group addresses to models, etc.

Note that smartphones currently do not support Bluetooth mesh directly so at least one device should be configured as a GATT Proxy to allow a smartphone to do configuration once provisioning is done. The only alternative currently is to have a gateway on the mesh network that allows the smartphone to access the mesh network indirectly.

7A.5 Cypress Mesh Helper Programs

There are several programs provided by Cypress that can be used for provisioning, configuration, and communication for mesh networks. There are:

- Host Applications (these communicate via an additional kit connected to the computer over the HCI UART – USB interface which then communicates with the Mesh network using the BLE radio in the additional kit):
 - Client Control Mesh (Windows)
 - Mesh Client (Windows, MacOS, Linux)
- Peer Applications (these communicate with the MESH kit directly using the BLE radio of the computer/phone):
 - Mesh Client (Windows 10)
 - Android application
 - iOS application

More information on these apps can be found here:

<https://www.cypress.com/documentation/software-and-drivers/bluetooth-mesh-helper-applications>

These applications are in a library that must be included as dependencies in the application in order for them to be available. The library is:

BTSDK Host and Peer Apps for Mesh (btsdk-host-peer-apps-mesh)

7A.5.1 Client Control Mesh Host App (Windows Only)

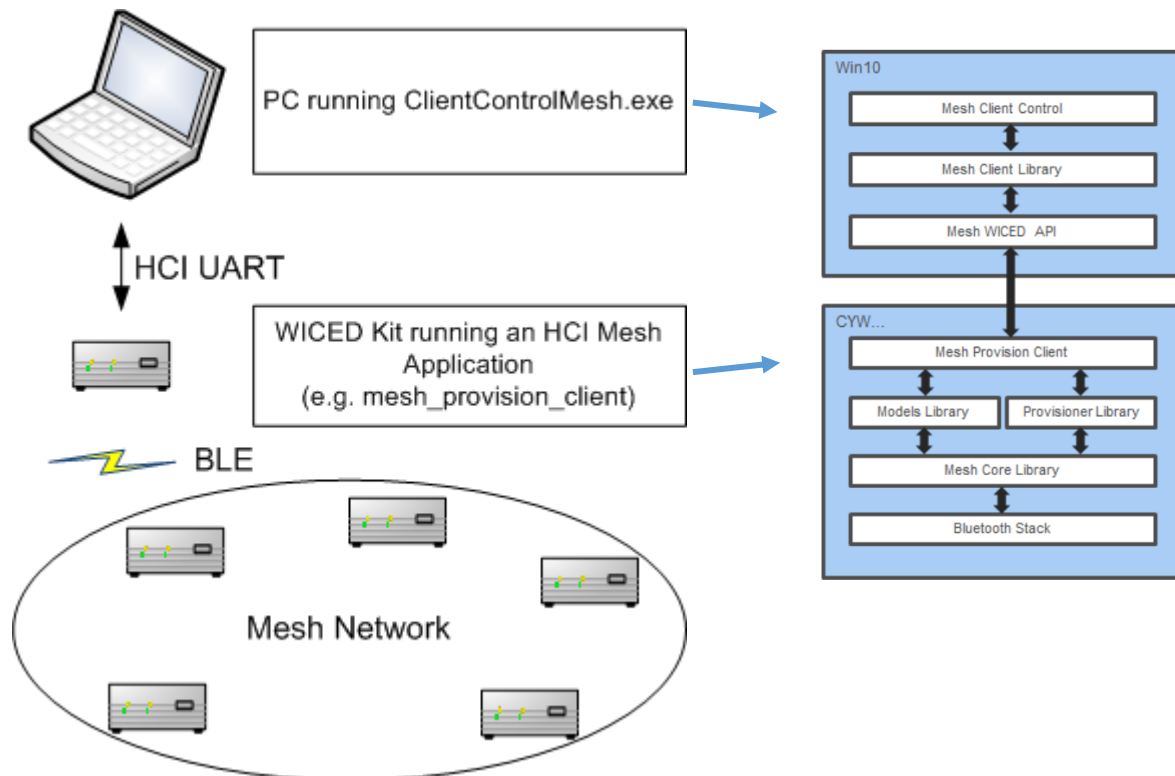
This application is an extension of the Client Control application that can be used for other WICED Bluetooth applications. A link to launch ClientControlMesh can be found in the Tools section of the Quick Panel (assuming that it is included in the `CY_BT_APPS_TOOLS` variable in the *Makefile*).

The source code and executable are available in the `wiced_btsdk` at:

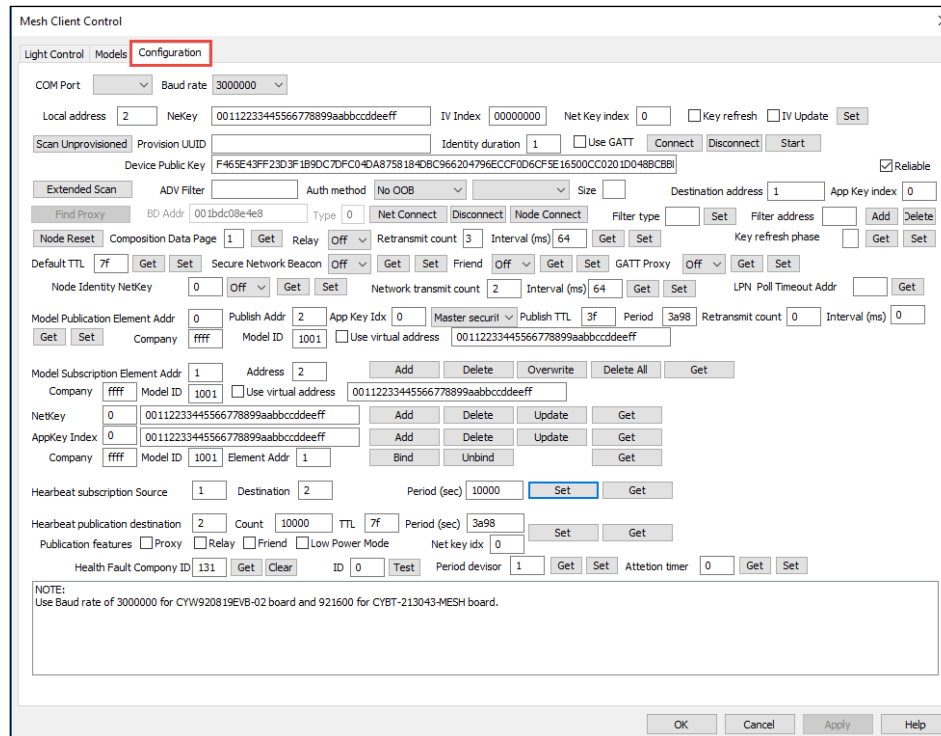
`wiced_btsdk/tools/btsdk-host-peer-apps-mesh/<version>/host/VS_ClientControl`

(The executable is under the *Release* directory)

To use it, you must first program a kit with an application that accepts HCI commands from the PC and then translates them to the appropriate Mesh commands to send out over the radio to the Mesh network. For example, the "Mesh Snip Provision Client" application supports a provisioning client, OnOff client, level client, light lightness client, etc. That kit can receive commands from a PC over the HCI UART and will in turn send the appropriate messages to the Mesh network.



The application has tabs for Configuration (provisioning, etc.), Models (for interacting with any model), and Light Control (specific for lighting applications). It allows very low-level interaction with the mesh network as you can see in the screenshots below. Note that if you put the helper application (e.g. Mesh Snip Provision Client) on a CYW920819EVB-02 kit, the default HCI baud rate is 3000000 while on a CYBT-213043-MESH kit it is 921600.



The screenshot shows the 'Mesh Client Control' application window with the 'Configuration' tab selected. The interface includes various input fields and buttons for configuring the mesh network. Key settings visible include:

- COM Port:** A dropdown menu.
- Baud rate:** Set to 3000000.
- Local address:** Set to 2.
- NetKey:** 00112233445566778899aabbccddeeff.
- IV Index:** 00000000.
- Net Key index:** 0.
- Key refresh:** Unchecked.
- IV Update:** Unchecked.
- Scan Unprovisioned:** A button.
- Provision UUID:** A text field.
- Identity duration:** 1.
- Use GATT:** Unchecked.
- Connect, Disconnect, Start:** Buttons.
- Device Public Key:** F465E43FP23D3F1B9DC7DFC04DA8758184DBC966204796ECCF006CF5E16500CC0201D048BCB8.
- Reliable:** Checked.
- Extended Scan:** A button.
- ADV Filter:** A text field.
- Auth method:** No OOB.
- Size:** A text field.
- Destination address:** 1.
- App Key index:** 0.
- Find Proxy:** A button.
- BD Addr:** 001bdc08e4e8.
- Type:** 0.
- Net Connect, Disconnect, Node Connect:** Buttons.
- Filter type:** A dropdown menu.
- Filter address:** A text field.
- Add, Delete:** Buttons.
- Node Reset:** A button.
- Composition Data Page:** 1.
- Relay:** Off.
- Retransmit count:** 3.
- Interval (ms):** 64.
- Get, Set:** Buttons.
- Key refresh phase:** A dropdown menu.
- Get, Set:** Buttons.
- Default TTL:** 7f.
- Get, Set:** Buttons.
- Secure Network Beacon:** Off.
- Get, Set:** Buttons.
- Friend:** Off.
- Get, Set:** Buttons.
- GATT Proxy:** Off.
- Get, Set:** Buttons.
- Node Identity NetKey:** 0.
- Off, Get, Set:** Buttons.
- Network transmit count:** 2.
- Interval (ms):** 64.
- Get, Set:** Buttons.
- LPN Poll Timeout Addr:** A text field.
- Get:** Button.
- Model Publication Element Addr:** 0.
- Publish Addr:** 2.
- App Key Idx:** 0.
- Master securit:** A dropdown menu.
- Publish TTL:** 3f.
- Period:** 3a98.
- Retransmit count:** 0.
- Interval (ms):** 0.
- Get, Set:** Buttons.
- Company:** ffff.
- Model ID:** 1001.
- Use virtual address:** Unchecked.
- 00112233445566778899aabbccddeeff:** A text field.
- Add, Delete, Overwrite, Delete All, Get:** Buttons.
- Model Subscription Element Addr:** 1.
- Address:** 2.
- Add, Delete, Overwrite, Delete All, Get:** Buttons.
- Company:** ffff.
- Model ID:** 1001.
- Use virtual address:** Unchecked.
- 00112233445566778899aabbccddeeff:** A text field.
- Add, Delete, Update, Get:** Buttons.
- NetKey:** 0.
- 00112233445566778899aabbccddeeff:** A text field.
- Add, Delete, Update, Get:** Buttons.
- AppKey Index:** 0.
- 00112233445566778899aabbccddeeff:** A text field.
- Add, Delete, Update, Get:** Buttons.
- Company:** ffff.
- Model ID:** 1001.
- Element Addr:** 1.
- Bind, Unbind, Get:** Buttons.
- Heartbeat subscription Source:** 1.
- Destination:** 2.
- Period (sec):** 10000.
- Set, Get:** Buttons.
- Heartbeat publication destination:** 2.
- Count:** 10000.
- TTL:** 7f.
- Period (sec):** 3a98.
- Set, Get:** Buttons.
- Publication features:** Proxy, Relay, Friend, Low Power Mode.
- Net key idx:** 0.
- Health Fault Company ID:** 131.
- Get, Clear, ID, Test, Period divisor:** 1.
- Get, Set, Attention timer:** 0.
- Get, Set:** Buttons.
- NOTE:** Use Baud rate of 3000000 for CYW920819EVB-02 board and 921600 for CYBT-213043-MESH board.
- OK, Cancel, Apply, Help:** Buttons.

Mesh Client Control

Light Control **Models** Configuration

COM Port Baud rate 3000000 OnOff ☒ Reliable ☐ Use publication info Destination address 1 App Key index 0

Battery Level Time to discharge Time to charge Prop ID Value

Local North Local East Altitude Global Latitude Longitude Altitude

Floor # Update time Precision ☐ Mobile Scene

On/Off Target Off Default Transition Time ☒ Use default transition time Scheduler Advanced

On Power Up State Transition Time 20000 Delay 25 1

Level Target Delta ☐ Continue Move Set Current 10/12/2019 5:21:57 PM

Lightness ☐ Linear Target Current Time Subsecs Uncertainty

Last Default Range Min -

Light HSL Hue Saturation

Light CTL Temperature Delta UV Default Get Default Set Temp Range -

Light xyl xyl x y Range x - y

Light LC Mode Occupancy mode Light On/Off

Sensor DESCRIPTOR GET Property ID Value Setting values : Property ID Raw Data

Cadence values : ☐ Trigger Type Fast Cadence Period Div Trigger Delta Trigger Delta Up Min Interval Fast Cadence High Low

CTL TTL: 00 DST: 0001 PDU: 05 00 00 50 73 20 0F 0F 0F 0F 0F 0F 0F 0F 0F 0F Net Send ☐ szmc Transp Send Addr: 1202 VS Data

Set Test Mode ☐ IV UPDT Transist 01 02 03

NOTE:
Use Baud rate of 3000000 for CYW920819EV8-02 board and 921600 for CYBT-213043-MESH board.

Mesh Client Control

Light Control Models Configuration

COM Port Baud rate 3000000

Application

Network User WIN-GJL

Current group

Group

Provision UUID

☐ Static public key

☐ Static OOB data

Identity duration 1

Device configuration

☒ Friend ☒ GATT Proxy ☒ Relay ☒ Net beacon Retransmit count 3 Interval (ms) 100 Default TTL 63 Network transmit count 3 Interval (ms) 100

Publish period Master securit Publish TTL 63 Retransmit count 0 Interval (ms) 500

Rename New name

Move Device from to group

Use Device

On/Off

Level

Lightness

Lightness Hue Saturation

Lightness Color Temperatur Delta UV

Vendor data Sensor Light Controller

NOTE:
Use Baud rate of 3000000 for CYW920819EV8-02 board and 921600 for CYBT-213043-MESH board.

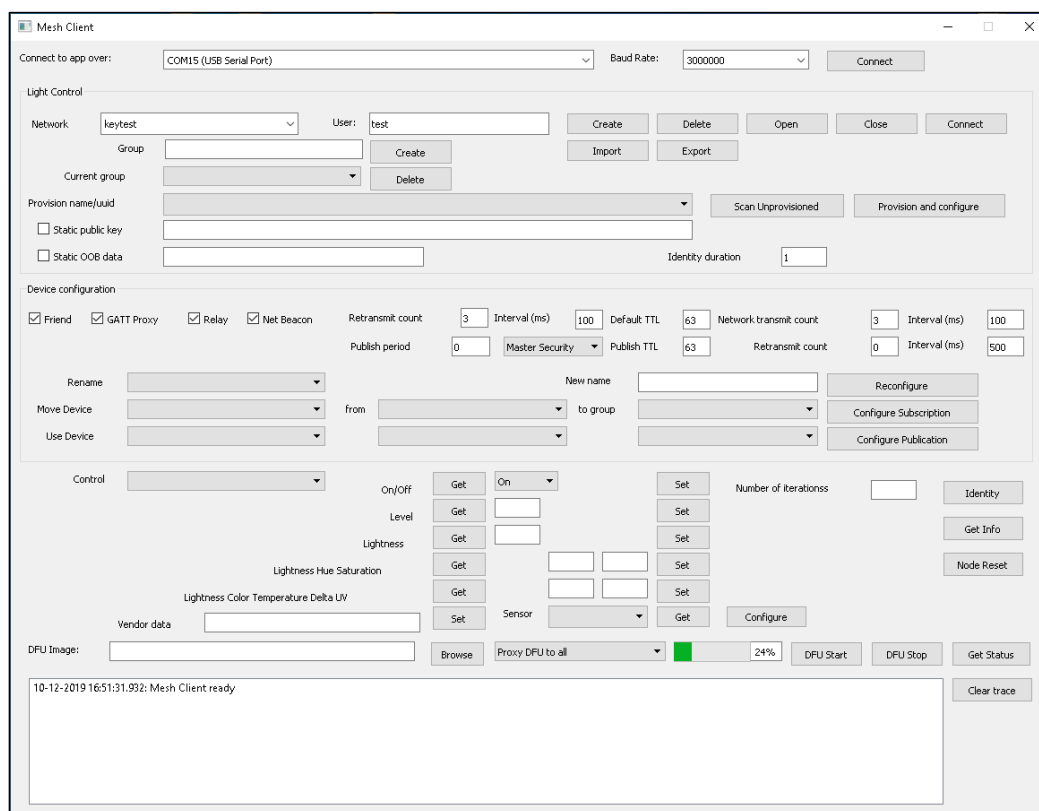
7A.5.2 Mesh Client Host App (Windows, MacOS, Linux)

As with the ClientControlMesh application, this application uses the HCI interface to talk to a kit containing BLE Mesh functionality. That kit will communicate over the BLE network to the provision and control a kit with a Mesh application programmed onto it.

This application doesn't show up in the Quick Panel but the executable can be launched from:

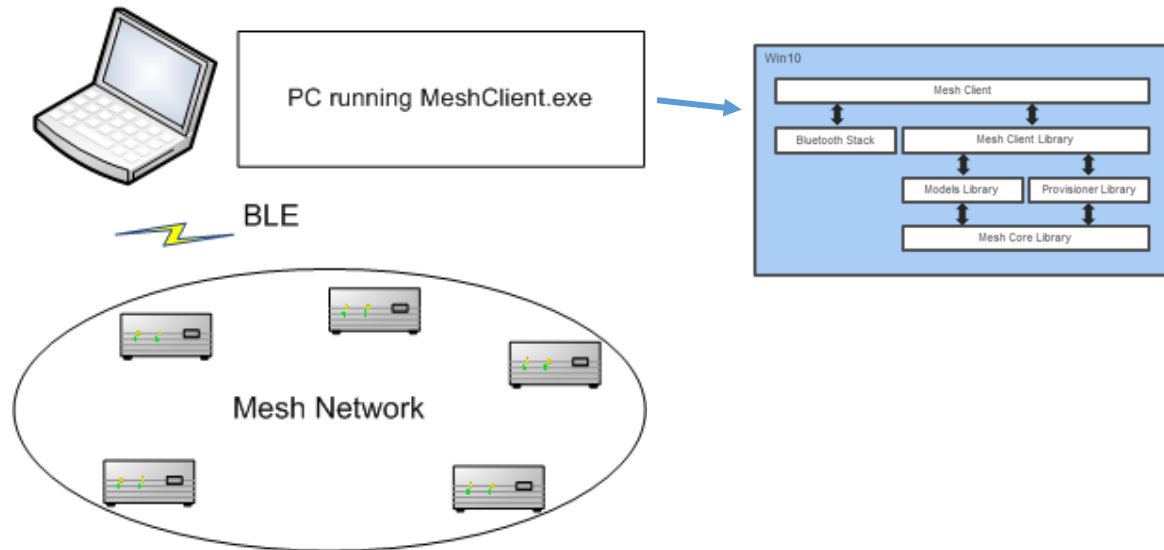
```
wiced_btsdk/tools/btsdk-host-peer-apps-mesh/<version>/host  
/Qt_ClientControl/<windows|Linux64|macos>/mesh_client
```

Unlike ClientControlMesh, this application has only a single tab that allows provisioning, general Mesh network control, and lighting control. The upside is that it will run on Windows, MacOS and Linux platforms.



7A.5.3 Mesh Client Peer App (Windows 10 only)

This application communicates with the specified mesh network directly using the BLE radio of the computer.

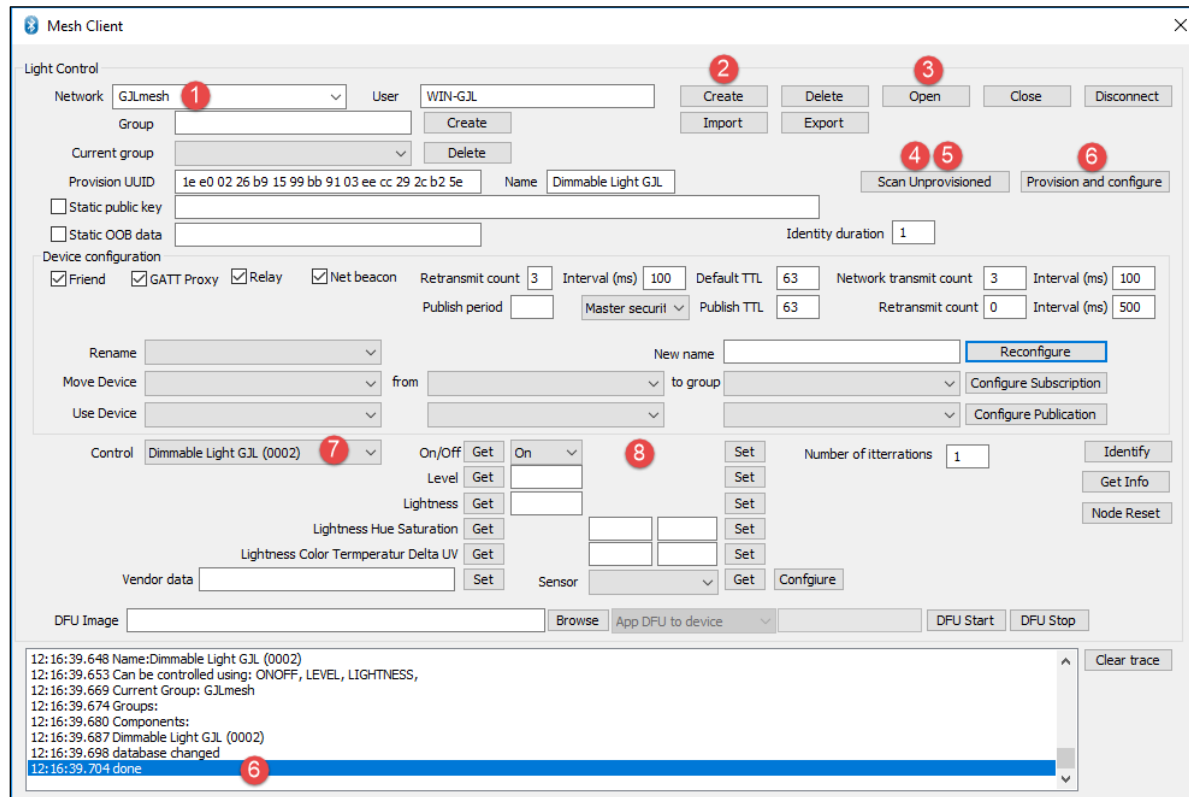


Note that support for BLE was added in Windows 10 so you can't use this with earlier versions of Windows. It can create mesh networks, provision devices and control lighting devices. This tool can be launched from the Quick Panel but it is not included in most template applications by default. To get it to appear in the Quick Panel, add `MeshClient` to the `CY_BT_APP_TOOLS` variable in the *Makefile* and then click **Refresh Quick Panel** in the Quick Panel.

The executable and source code can be found at:

`wiced_btsdk/tools/btsdk-host-peer-apps-mesh/<version>/peer/Windows/MeshClient`

(The executable is under the *Release* directory)



The basic flow for using the application is:

1. Enter a name for your network
2. Click **Create**.
3. Click **Open**.
4. Click **Scan Unprovisioned**.
5. Wait a few seconds (5-10) for devices to appear in the list and click **Stop Scanning**.
 - a. Click the down-arrow to get a list of all unprovisioned devices that have been found.
 - b. If you don't see your device in the list, scan for a longer period of time until you see the device you are looking for.
6. Click **Provision and configure**.
 - a. This step will take a few seconds – wait until it is complete before continuing.
7. Select your device in the **Control** dropdown.
8. Use **On/Off Get**, **On/Off Set**, **Level Get**, **Level Set**, etc. to control your device.

7A.5.4 Cypress Mesh Controller App (Android)

The Cypress Android app is provided with the BTSDK. Source code is available in the SDK for those who want an example to create their own custom Android mesh app.

The app communicates with the mesh network using the device's BLE capabilities. Since smartphones don't (yet) have mesh capability, the app uses GATT connections for provisioning and relies on the presence of a GATT proxy for mesh configuration and communication.

The app can create mesh networks, provision, configure, and control devices. It can be found on the Google Play store by searching for: "Cypress MeshAPP".

Alternately, the installable file is located at:

```
wiced_btstack/tools/btstack-host-peer-apps-  
mesh/<version>/peer/Android/src/bin/MeshController.apk
```

You can install the app manually by dragging the .apk file onto the phone's filesystem and then executing it to install the app. You will need to allow installation of 3rd party applications for this to work. The SDK also includes the source code which you can use as a basis for creating your own Android app.

The basic flow for using the application is:

1. Create a network.
2. Add or select a group (ALL is created by default).
3. Add a device to the group.
 - a. This will take a few seconds. Wait until it completes before proceeding
4. Connect to the network if it doesn't happen automatically (bar will be green when connected)

Note You must have a GATT proxy provisioned on the network to connect to it.

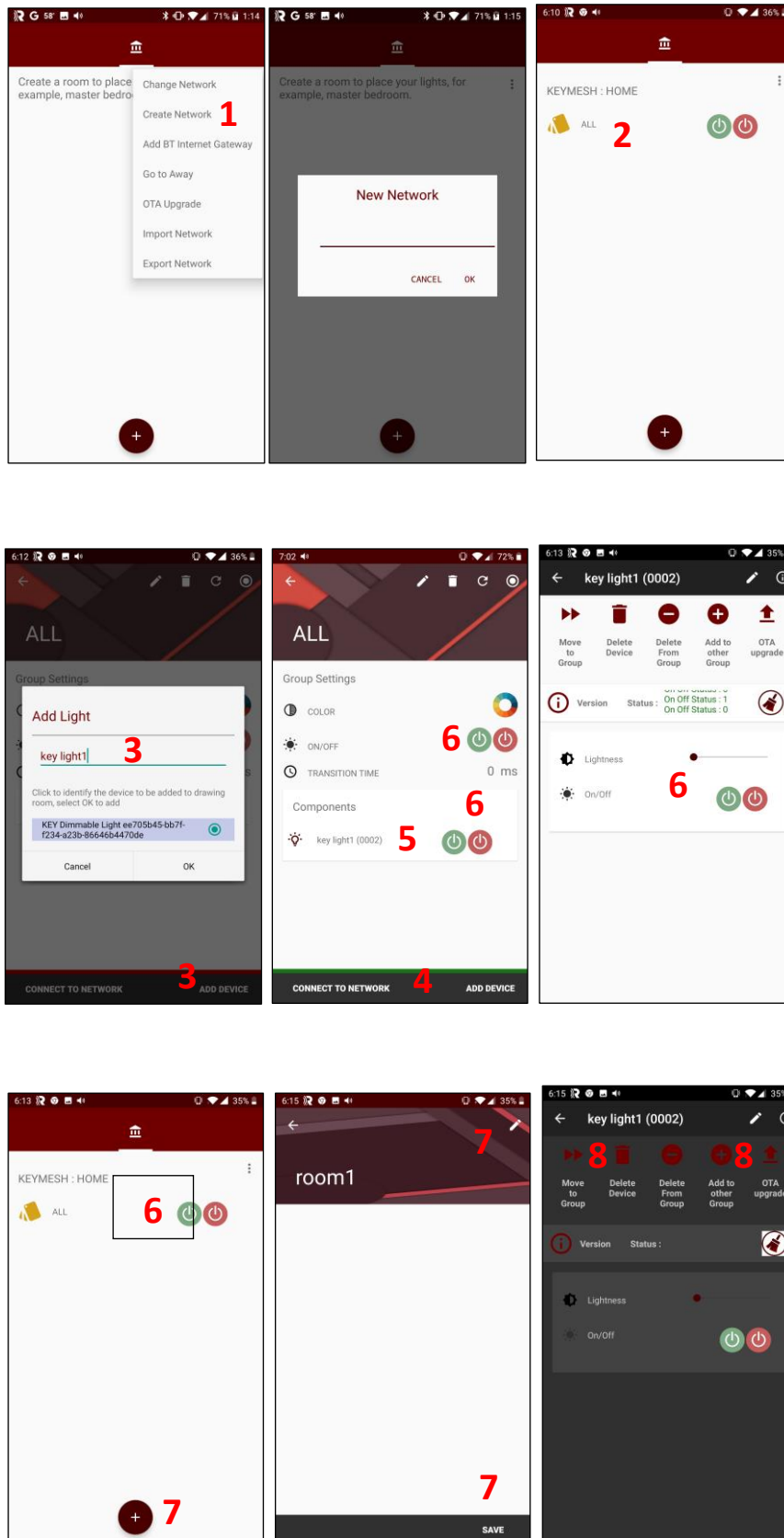
Note As soon as you provision a device with a GATT proxy, the app will connect to it. If you then provision a device that doesn't have a GATT proxy, it will not re-connect so you will have to re-connect manually using the button.

5. Select the device.
6. Control the device.

Note You can control all devices simultaneously at the group level or individually at the device level.

Note At the device level, you can adjust the lightness in addition to the on/off control.

7. Optional: Click the + to add additional Groups (i.e. Rooms).
8. Optional: Move or Add devices to other Groups.



7A.5.5 Cypress Mesh App (iOS)

The Cypress iOS app is similar to the Android app. The app communicates with the mesh network using the device's BLE capabilities. Since smart phones don't (yet) have mesh capability, the app uses GATT connections for provisioning and relies on the presence of a GATT proxy for mesh configuration and communication. The app can create mesh networks, provision/configure devices and can control lighting devices. The source code for the application is located at:

`wiced_btstack/tools/btstack-host-peer-apps-mesh/<version>/peer/iOS/MeshApp`

This app is on the Apple App Store under the name *Cypress MeshApp*, a link to the app store page can be found here:

<https://www.cypress.com/documentation/software-and-drivers/bluetooth-mesh-helper-applications>

The basic flow for using the application is similar to the Android app:

1. Login or create an account

The account is local to the device – it is just used to identify different users on that device. If you re-install the app or delete the app's data, you will need to recreate the account.

2. Create a network
3. Add a group (ALL is created by default) and select it
4. Add a device to the group

This will take a few seconds. Wait until it completes before proceeding

5. Connect to the network if it doesn't happen automatically

Note You must have a GATT proxy provisioned on the network to connect to it.

Note As soon as you provision a device with a GATT proxy, the app will connect to it. If you then provision a device that doesn't have a GATT proxy, it will not re-connect so you will have to re-connect manually using the button.

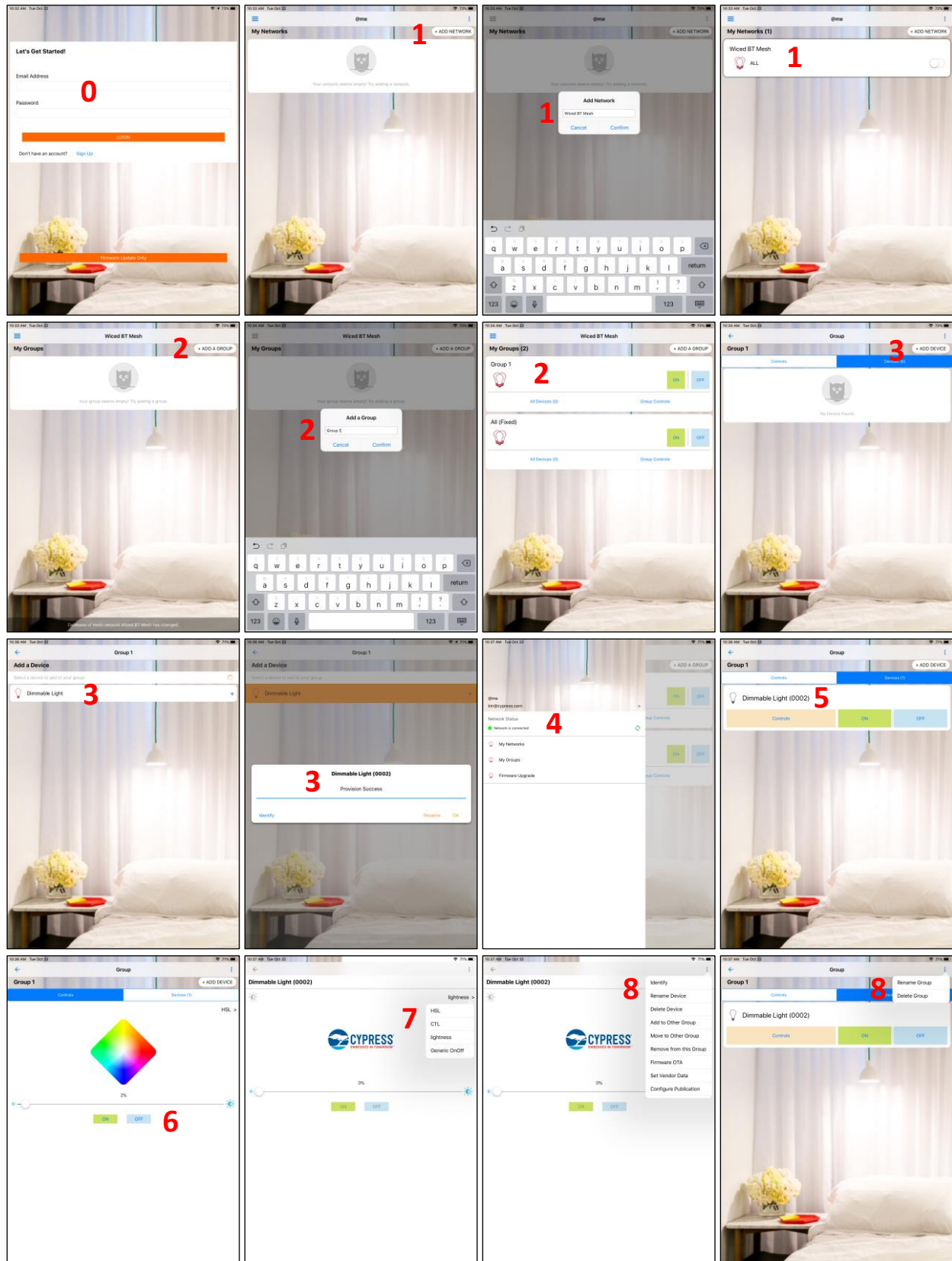
6. Select the device
7. Control the device

Note You can control all devices simultaneously at the group level or individually at the device level.

Note At the device level, you can adjust the lightness in addition to the on/off control.

8. Optional: You can change the light control display using the drop down at the Device level. If you have a device that includes a "Light Lightness Model" you should select "lightness" from the drop down to get the correct controls.
9. Optional: Use the menus at the Group and Device level to rename groups, move or add devices to other groups, etc.

Note As with individual device controls, you should pick the correct type of control for the devices in your group from the drop down.



7A.6 Demo

A mesh network with several lights and rooms will be demonstrated to the class at this point.

7A.7 Exercises

Exercise 7A.1 Create Network with a Dimmable Light Mesh Device

In this exercise you will create your own (very small) mesh network.



1. Create a new application for the CYBT-213043-MESH kit.

Use the *Mesh Demo Dimmable Light* template application and name it **ch07a_ex01_light_dimmable**.



2. Open the file *light_dimmable.c* and find the `device_name`.

Change the name so that it has your initials in it (e.g. "<Inits> Dimmable Light").



3. Program the application to one of the CYBT-213043-MESH kits.

Hint You should open a terminal window to the PUART to see messages. **The default PUART baud rate for the mesh applications is 921600.**

- **Hint** If your terminal emulator does not support 921600, open the file in *wiced_btstack/dev-kit/libraries/btstack-mesh/<version>/COMPONENT_mesh_app_lib/mesh_app_hci.c* and search for the second occurrence of 921600. Change that value to one that is supported and rebuild/reprogram.
- **Hint** If you change that value, remember that it is in a library that is shared so it will affect all MESH apps that use the same *mtb_shared* location.



4. Run the Mesh Lighting application to provision the device.

Hint If you don't see any devices listed after ~10 seconds, exit the app, stop/restart BLE and then try again.