

MeshClient and ClientControlMesh App User Guide

ModusToolbox

About this document

Scope and purpose

This document provides quick start instructions for the MeshClient and the ClientControlMesh applications, which are part of WICED® Studio and ModusToolbox®.

Intended audience

This document is intended for embedded application developers using WICED Studio or ModusToolbox to implement and test Mesh-based solutions with WICED Bluetooth devices.

Table of contents

About this document.....	1
Table of contents.....	1
1 Introduction	3
1.1 Acronyms and Abbreviations	3
1.2 IoT Resources and Technical Support.....	3
2 Overview.....	4
2.1 Mesh Libraries.....	6
3 MeshClient Applications Overview	7
3.1 Provisioning.....	7
3.2 Configuration.....	7
3.3 Control.....	7
4 Using the MeshClient Application	9
4.1 Creating and Opening a Mesh Network.....	9
4.1 Adding a Node	11
4.2 Creating and Managing Groups	15
4.3 Configuring Devices.....	17
4.4 Over-The-Air Device Firmware Upgrade.....	18
5 Mesh Performance Testing	20
5.1 Overview	20
5.2 Key Performance Indicators	20
5.2.1 Probability/Reliability	20
5.2.2 Latency	21
5.2.3 Number of Hops/Time-To-Live (TTL)	21
5.2.4 Power Consumption	21
5.2.5 Packet Length/Payload Size	22
5.2.6 Network/Relay Retransmission	22
5.2.7 Methods to Measure Performance Indicators.....	23

5.2.8	Direct/One-way Approach	23
5.2.9	Indirect/Round-trip Approach	24
Figure 5 Indirect/Round-trip Approach.....		25
5.3	Testing Procedure	25
5.3.1	Application Settings and Configuration	31
References.....		33
Revision history.....		34

1 Introduction

1.1 Acronyms and Abbreviations

In most cases, acronyms and abbreviations are defined on first use. For a comprehensive list of acronyms and other terms used in the documents, go to the [Glossary](#).

1.2 IoT Resources and Technical Support

The wealth of data available [here](#) will help you to select the right IoT device for your design, and quickly and effectively integrate the device into your design. You can access a wide range of information, including technical documentation, schematic diagrams, product bill of materials, PCB layout information, and software updates. You can acquire technical documentation and software from the [Support Community website](#).

2 Overview

Bluetooth SDK in WICED Studio and ModusToolbox offer wide variety of Bluetooth SIG Mesh 1.0-related products. One of them is a set of portable libraries that can be used on any platform to create an application to provision and control the mesh. WICED Studio and ModusToolbox support Bluetooth Mesh on CYW20706, CYW20735, CYW20719, CYW20819, and Cypress modules such as CYBT-213043-02 based on these silicon devices.

The MeshClient and the ClientControlMesh applications provide a sample Windows implementation that show how to use interfaces exposed by the mesh libraries. The MeshClient works only with Windows 10. The MeshClient application uses PC's built-in Bluetooth radio, or an external Bluetooth dongle to communicate with Bluetooth mesh. The MeshClient application implements all layers of the mesh stack. The ClientControlMesh application implements only the application layer. It uses the Mesh Models and Mesh Core libraries residing on the embedded device that requires a Cypress device to act as a client and hence requires an extra evaluation board to be connected to the PC for mesh operation. Any of the Cypress devices that support Bluetooth mesh can be used for this application irrespective of the device used by the mesh nodes. The ClientControlMesh can be used with any version of Windows operating system.

The MeshClient and the ClientControlMesh Windows applications are installed with WICED Studio and ModusToolbox installation as part of Bluetooth SDK.

- **App paths in WICED Studio** (Change 6.4 to appropriate version based on the WICED Studio version being used)

If the default path for the installation is used, the MeshClient project is in:

`C:\Users\<user>\Documents\WICED-Studio-6.4\common\apps\snip\mesh\peerapps\Windows\MeshClient`

To open the application on Windows machine:

- Go to: `C:\Users\<user>\Documents\WICED-Studio-6.4\common\apps\snip\mesh\peerapps\Windows\MeshClient\Release\x86`
- Double-click the **MeshClient** application.

The Windows **ClientControlMesh** project is in:

`C:\Users\<user>\Documents\WICED-Studio-6.4\common\apps\snip\mesh\VS_ClientControl`

To open the application on a Windows machine:

- Go to: `C:\Users\<user>\Documents\WICED-Studio-6.4\common\apps\snip\mesh\VS_ClientControl\Release.`
- Double-click the **ClientControlMesh** application.

A version of the **ClientControlMesh** project is provided for Linux and OSX and can be found in:

`C:\Users\<user>\Documents\WICED-Studio-6.4\common\apps\snip\mesh\Qt_ClientControl`

- **App paths in ModusToolbox**

MeshClient and **ClientControlMesh** are provided in the `wiced_btstack` project in the ModusToolbox IDE, which is created and used by any WICED Bluetooth application created in the IDE.

The **MeshClient** project (Windows only) can be found in the IDE Project Explorer pane in:

`mtb_shared\wiced_btstack\tools\btstack-peer-apps-mesh\Windows\MeshClient\Release\x86`

Overview

Similarly, the Windows **ClientControlMesh** project is in:

mtb_shared/wiced_btsdk/tools/btsdk-host-apps-mesh/VS_ClientControl

A version of the **ClientControlMesh** project is provided for Linux and OSX and can be found in:

mtb_shared/wiced_btsdk/tools/btsdk-host-apps-mesh/Qt_ClientControl

To open the applications, select the embedded Mesh application project in the ModusToolbox IDE Project Explorer pane, then click the appropriate tool execution link from the Quick Panel.

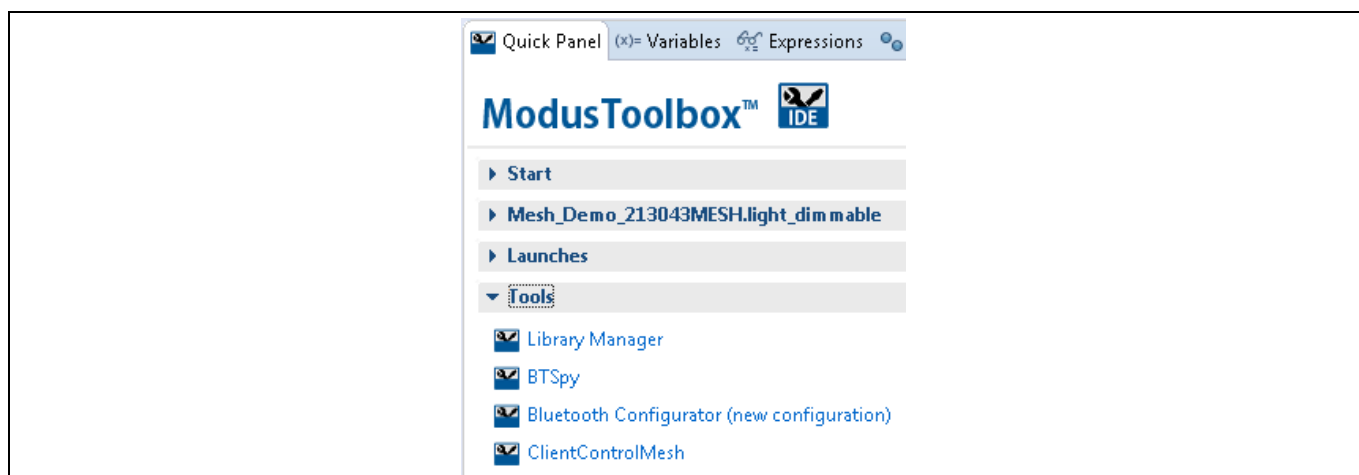


Figure 1 Tools in Quick Panel

Alternately, navigate to the workspace location in the file system, and locate the executable under the appropriate path based on the Project Explorer locations listed above, and double click to run the programs.

The Windows ClientControlMesh and the MeshClient applications can be built using Microsoft Visual Studio 2019 or later release.

Operating System (OS) Requirements

- **MeshClient** - The MeshClient application relies upon the Windows Bluetooth stack version which is available only in Windows 10 Creators Update. While launching the application, if you see the error “This application requires Windows 10 Creator Updates”, install these updates to run this application. Go to the following link click **Update now** to download and install these updates:
<https://www.microsoft.com/en-us/software-download/windows10>
- **ClientControlMesh** - The ClientControlMesh does not use Windows stack and can be executed on any version of Windows OS.

Figure 2 shows the software block diagram of the MeshClient (left) and ClientControlMesh (right) applications.

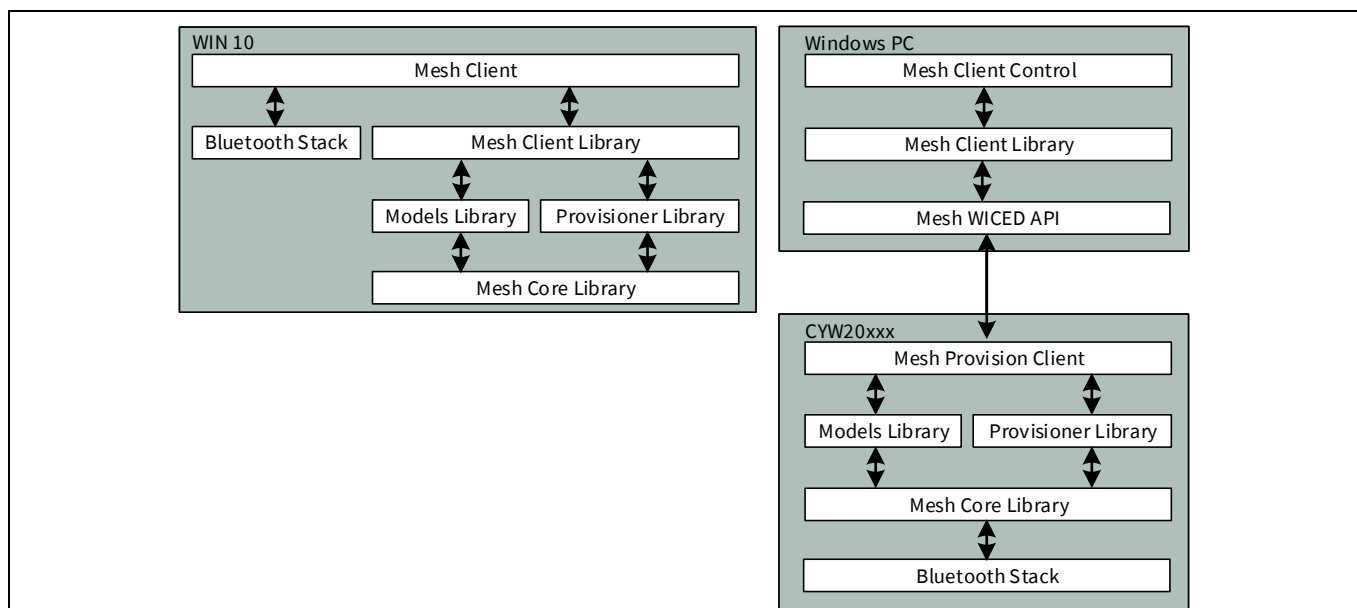


Figure 2 MeshClient (Left) and ClientControlMesh (Right) Software Block Diagram

The MeshClient application uses Bluetooth stack as it exists on Windows 10 OS. It uses GATT Proxy connection [1] to control the mesh. The ClientControlMesh application uses Bluetooth stack of the Cypress silicon. It can support both GATT Proxy and advertising channel to provision and control mesh devices.

The MeshClient and ClientControlMesh applications expose the functionality of various client models defined in the Mesh specifications including Configuration, Health, Default Transition Time, OnOff, Level, Power OnOff, Light Lightness, Light HSL, Light CTL, Sensor, and a sample Vendor-Specific client models. Other client and server models can be added in future releases.

2.1 Mesh Libraries

The MeshClient library executes the state machines required for provisioning and configuration. It provides an interface to the application to test the mesh functionality. The library maintains the database for the mesh network.

In the MeshClient application, to execute Bluetooth functionality such as starting BLE scan, establishing a connection to a specific device, or sending a data packet, the MeshClient library executes methods that are provided by the MeshClient application, which in turn uses Bluetooth Stack of the OS. On the other hand, in the ClientControlMesh application, the MeshClient library uses Mesh WICED API to control embedded application to perform all the mesh related work.

The Mesh Core, Models, Provisioner and Core libraries implement all the functionality as defined in the Bluetooth SIG Mesh Profile [1] and Mesh Models [2] specifications.

3 MeshClient Applications Overview

3.1 Provisioning

Provisioning is a process of adding new nodes into a Mesh network. Provisioning is performed by a special node called a “Provisioner”. The MeshClient/ClientControlMesh applications perform as a Provisioner in the mesh network. MeshClient and ClientControlMesh applications maintain the database for the network, initiate the scan for unprovisioned devices, and perform the provisioning procedure as defined in Mesh Profile specification [1]. As a result of execution of the provisioning procedure, the Provisioner provides to the new node a bare minimum of the information to be a part of the mesh network (network key, IV index) and establishes the Device Key for the new node that is used between the Provisioner and the node during the configuration stage.

While the Mesh specification allows provisioning over the Advertising (PB-ADV) and GATT (PB-GATT) bearers, the MeshClient uses the GATT bearer only because it relies on the Microsoft Bluetooth stack as transport. The MeshClient Control can be configured to use any bearer.

3.2 Configuration

It is not enough just to provision a device to make it a fully functional mesh node of the mesh network. The following is a partial list of things that the Provisioner needs to perform during the configuration:

- Read the new node’s composition data to find out the device capabilities. For example, based on the information in the composition data, the Provisioner can figure out if it is a switch, a light bulb, or some other device.
- Set up the features that the new node should support. For example, if the node supports GATT Proxy or Friend role, the Provisioner needs to specify if the node should use the feature.
- Add network keys if the node should also be a part of other subnets, and add application keys for use with various models.
- Bind appropriate application keys to appropriate models of the new node. For example, the Provisioner can specify one application key to be used to configure the bulb and different application key to control the bulb.
- Configure various network parameters. For example, the Provisioner can specify the number of times the node should retransmit the message if it performs as a relay, and the number of times and frequency at which the node should publish the status messages.
- Configure the new device to be a part of a group.
- Configure clients, for example on/off switch, to control a specific server such as a light bulb, or a group of servers such as all light bulbs in a room.

3.3 Control

After the new node has been provisioned and configured, it can send and receive messages to and from devices in the same mesh network. For example, when you provision and configure a switch, the switch can send ON/OFF commands to a bulb or to all bulbs in the room.

The MeshClient and ClientControlMesh can act as various actuators including an on/off switch, a dimmer and a color control. For that purpose, they support corresponding client models and can send various Get/Set commands to control mesh devices. For example, the application can send a command to dim the light bulb to a certain level, or to adjust the color temperature of the light bulb.

MeshClient Applications Overview

Similar to any other client, the application can send messages to a single device or to a group of the devices. The replies are typically received from each device. When the application addresses the group with acknowledged message, each device in the group would send a reply. The mesh stack monitors how many replies have been received, and if reply from some specific node is not received, the Device Unreachable message is sent to the application.

Depending on the type of the device, some devices may act purely as clients, others like servers, and some can act simultaneously as client and servers. A simple generic ON/OFF switch is an example of a clean client. An HSL light bulb is an example of a pure server. There can be a node which is wired to two bulbs. There can be a power strip with one switch and several outlets, and the switch can be configured to control one of the outlets, or all outlets or the strip, or several strips.

4 Using the MeshClient Application

See Overview for the location and execution instructions for MeshClient and ClientControlMesh applications.

If a Windows 10 PC is used, it is recommended to use MeshClient application as it does not require an external device to run Bluetooth stack.

The user interface of the MeshClient and the ClientControlMesh applications are very similar. The only key difference between the two applications is COM port selection and Baud rate setting. These fields are not available in the MeshClient app as it uses PC's in-built Bluetooth. The ClientControlMesh application requires to talk to an external Cypress Evaluation board/device over the HCI UART. Hence these fields are provided in the ClientControlMesh application. See Using the MeshClient Application to learn how to select COM port and baud rate. Rest of this document uses screenshots from the MeshClient app as most of other fields and buttons are similar in both apps.

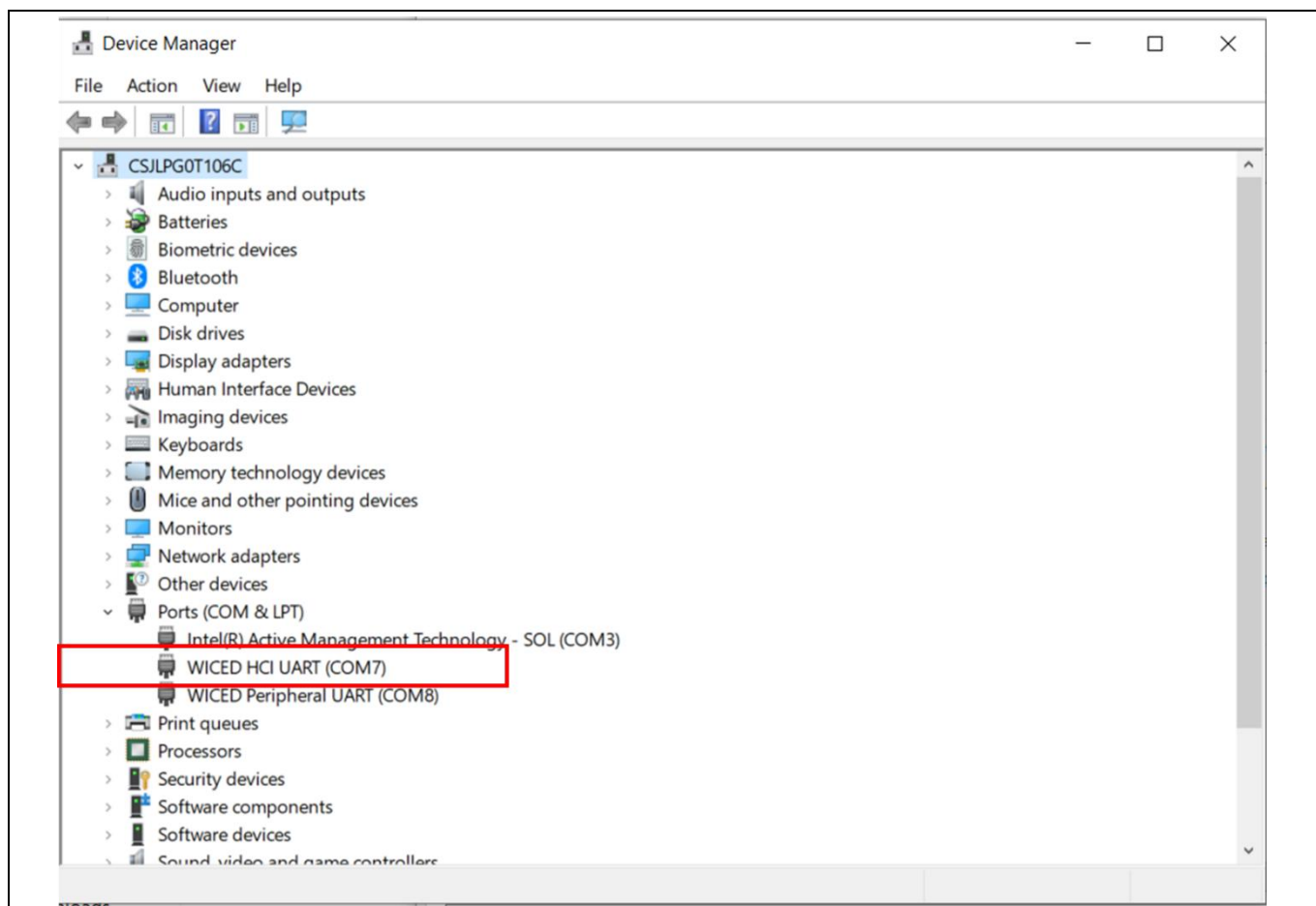
1.1 Creating and Opening a Mesh Network

Step 1: Jump to step 2 if you are using the MeshClient application. Continue here if using the ClientControlMesh application.

Program one evaluation board with mesh_provision_client snip application. This snip is available as part of WICED Studio installation under `apps\snip\mesh\mesh_provision_client`. If using WICED Studio, refer to the respective kit's user guide to learn how to build and download an application on to the board. If using ModusToolbox, see the Getting Started with Bluetooth Mesh Application Note [4] to learn how to download code examples from GitHub and program the board.

Once the board is programmed and connected to the PC, check the COM port number for the HCI UART. To check the COM port number, go to **Device Manager** on your PC and expand **Ports (COM & LPT)**. Here, look for **WICED HCI UART**. This is the COM port number to be used in the ClientControlMesh application. See the following screenshot.

*Note: If the PC is detecting HCI and UART ports as **USB Serial Port** without any distinction, then the lower COM port number is likely to be HCI UART's COM port number.*



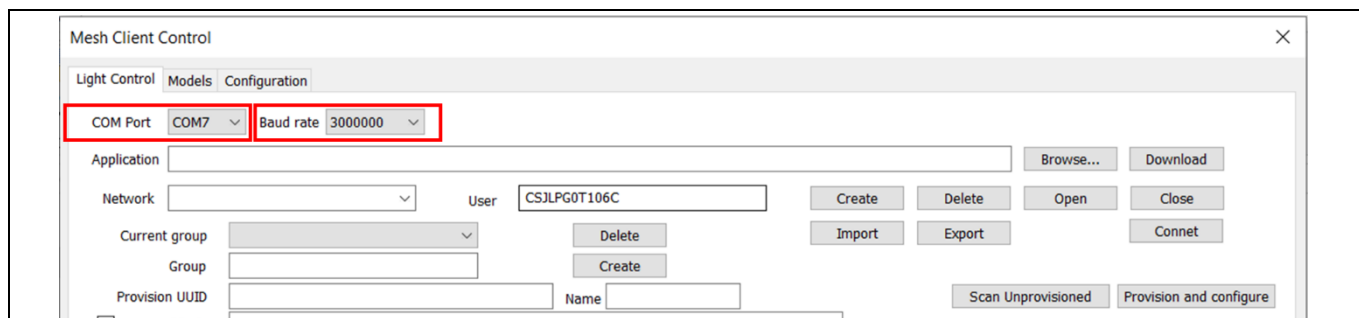
Once the COM port is identified, open the ClientControlMesh application.

Note: If the board is connected to PC after opening the ClientControlMesh application, the ClientControlMesh application will not detect the COM port. So, make sure you open application after the board is connected and enumerated.

Select this HCI-UART COM port from the **COM port** dropdown menu.

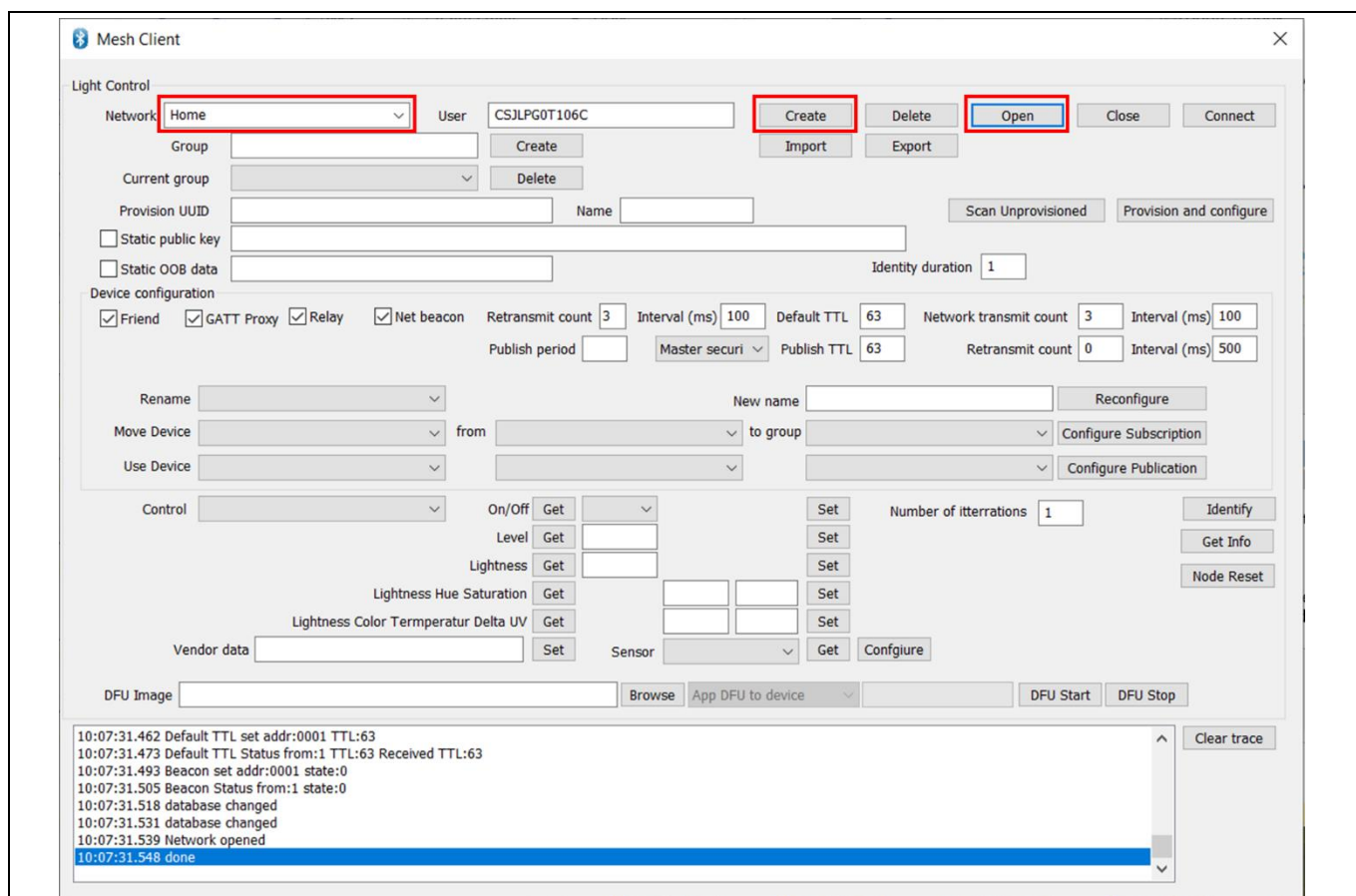
Then, select **Baud rate** 300000 if using CYW920819EVB-02 or any other chip-on-board evaluation boards.

Select **Baud rate** 115200 if the CYBT-213043-EVAL or CYBT-213043-MESH EZ-BT Mesh evaluation kits are being used to run provision control client.



Using the MeshClient Application

Step 2: In the **Network** field, type the string that you want to use as the network name. Click **Create** and then, click **Open**.



When a network is created, the MeshClient creates the required network attributes such as the mesh UUID, network and application keys, and saves the information in the mesh database which is stored in a JSON file in the directory where the application is started from. The schema of the Mesh Provisioner database is described in corresponding document from Bluetooth SIG [3].

There can be multiple networks controlled by the same PC, for example “Home”, “Office”, “Parent’s house”.

When you click **Open**, the MeshClient configures the stack with the parameters of the selected network. Similarly, the ClientControlMesh talks over the selected COM port to configure the stack running on the embedded platform. The “done” trace at the end of the configuration process indicates that the stack has been configured successfully.

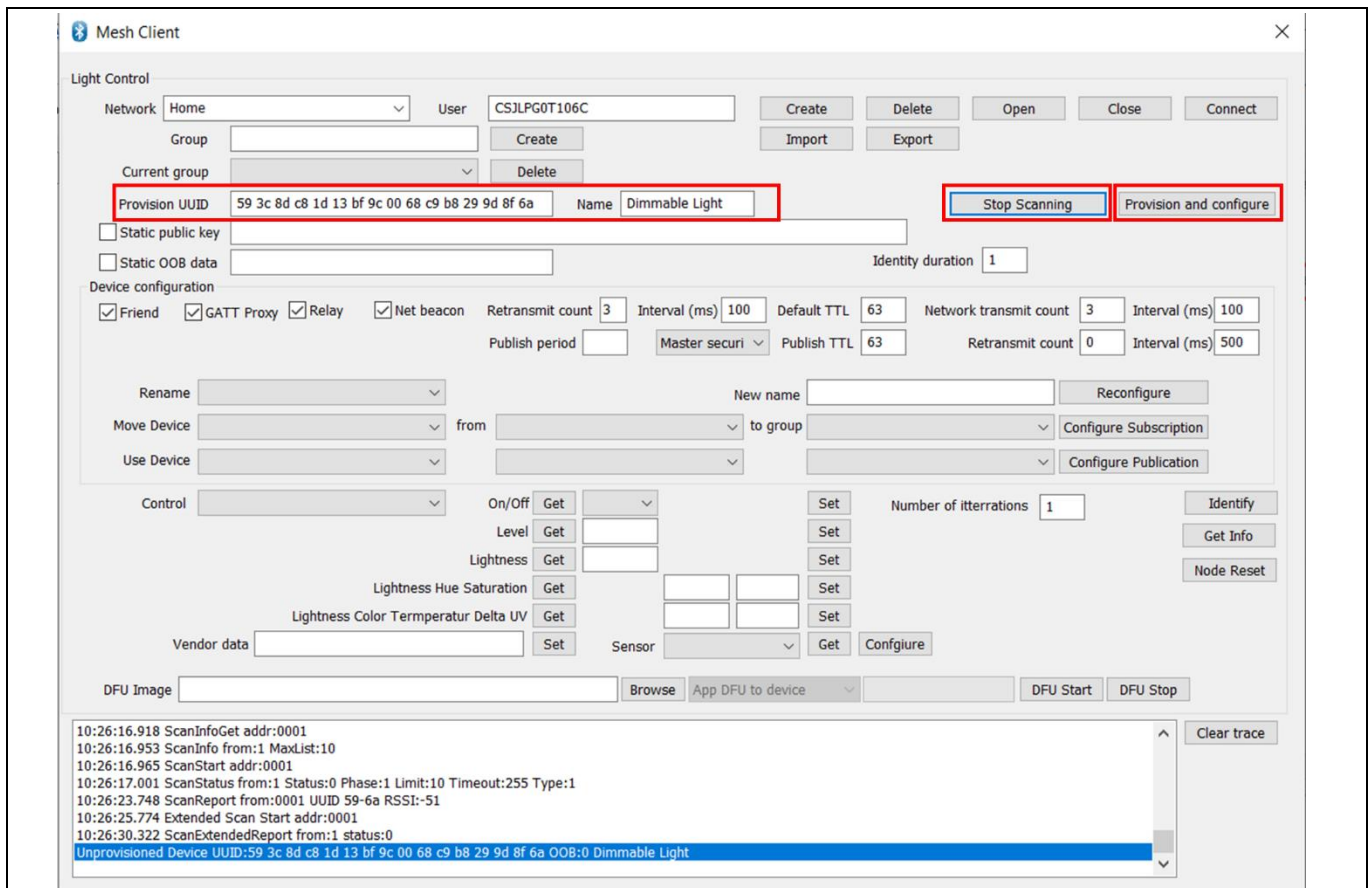
4.1 Adding a Node

Use WICED Studio or ModusToolbox to build and download one of the mesh samples to a WICED evaluation board. In the following description, the BLE_Mesh_LightDimmable code example is used. See the respective kits’ user guide/getting started guide to learn how to program the board.

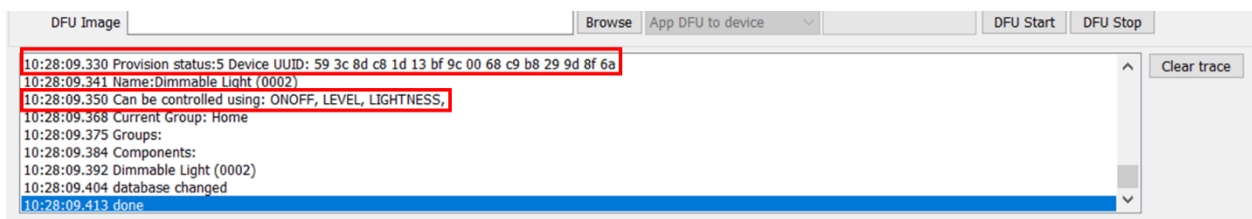
Using the MeshClient Application

Do the following to provision a new node, and see the following screenshot:

1. In the MeshClient window, click **Scan Unprovisioned**.
2. The title of the button changes to **Stop Scanning** to indicate that the scan is active.
3. The trace window displays the UUIDs of the devices that are in the radio range. The **Provisioned UUID** field is automatically filled with the UUID of the last discovered device.
4. When you see the device that you want to work with, click **Stop Scanning**.
5. Click **Provision and configure**.



The provisioning and configuration process consists of several steps. While the process is being executed, the status is displayed in the trace window. At the end of the process, **Provision status:5** appears in the trace window, indicating that the process has been completed successfully. The MeshClient also queries the library for the methods available for the application to control the device. For example, the provision device in the trace below can be controlled using On Off, Level as well as Lightness.

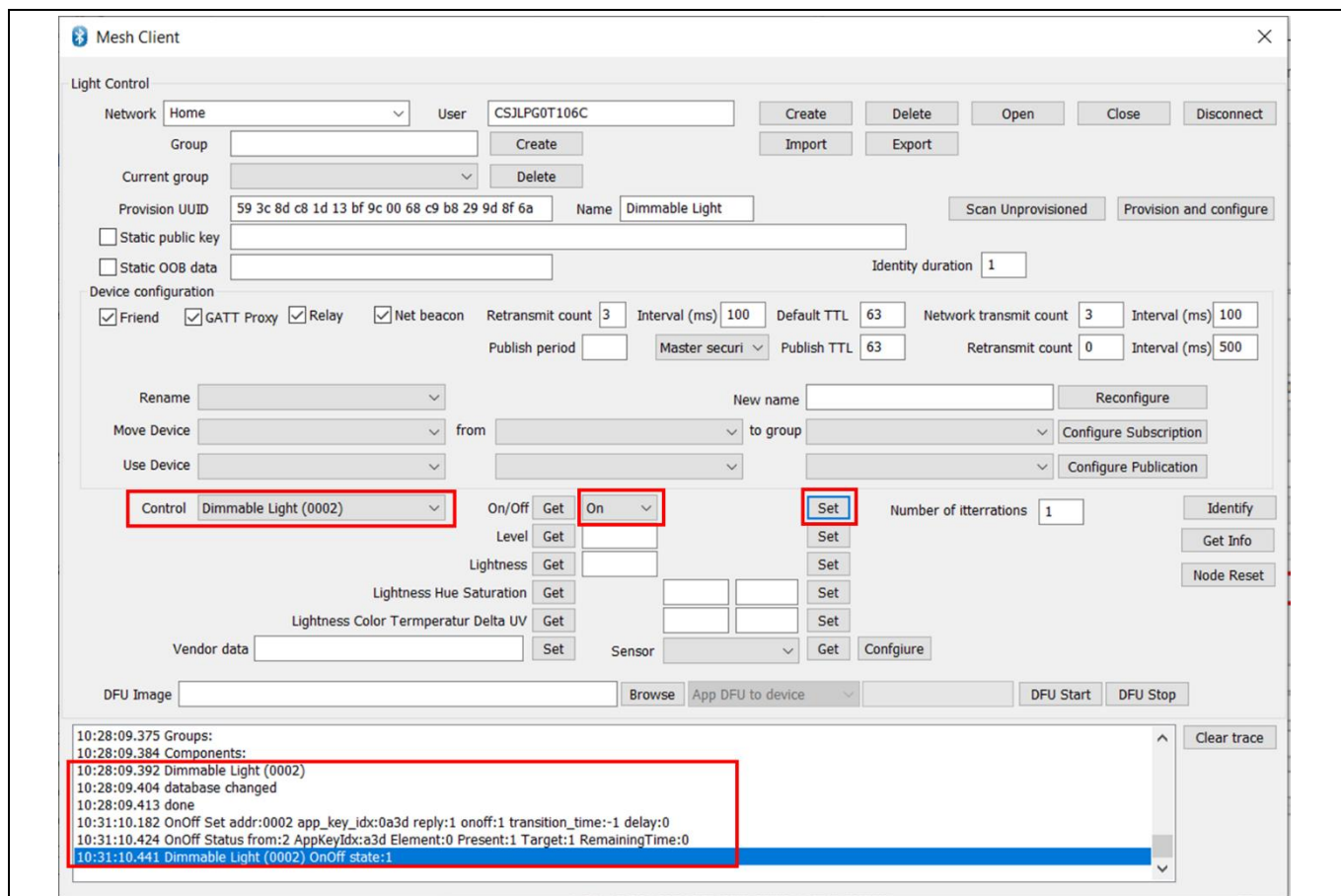


The trace window will print out the results of the operation.

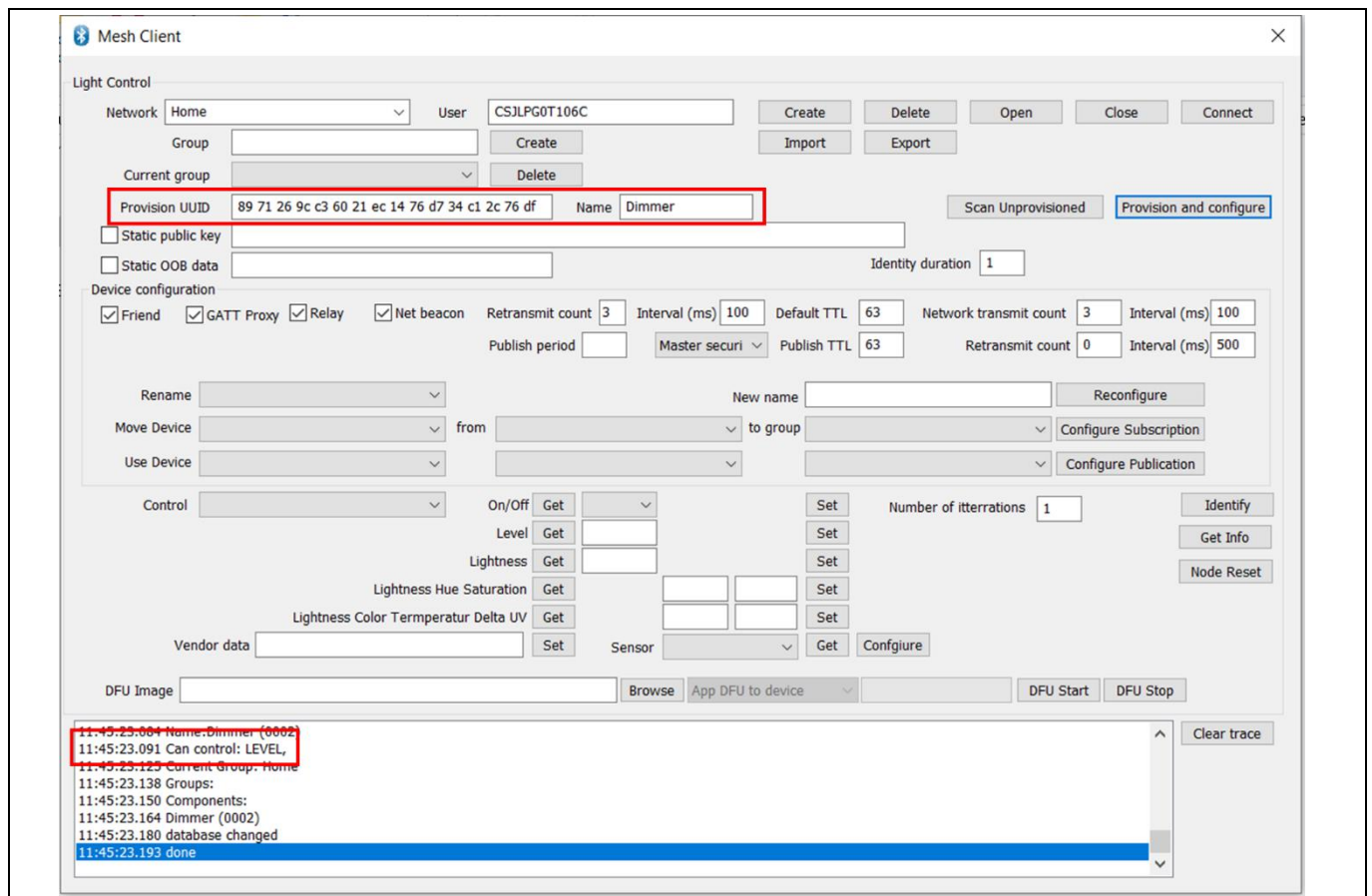
Using the MeshClient Application

If the device has been configured to be a GATT Proxy, the MeshClient will keep the connection to the new device open. If needed, click **Disconnect** to drop the GATT connection. When MeshClient is not connected to the mesh network, click **Connect** to establish the GATT connection.

The device is ready to use. For example, you can select the device and issue a **Get** the **On/Off status** command, or set a desired state to **On** and issue a **Set** command. Before sending any command to the node, ensure that app is connected to the proxy node i.e. the **Connect/Disconnect** button must shows **Disconnect**.



When a switch/dimmer or any other client is provisioned, instead of the “Can be controlled using” tag, the trace window will show “Can control”. For example, following screenshot shows the “Can Control” trace when a dimmer (level client model) is provisioned. As it is a level client, it can control level as shown in the traces.

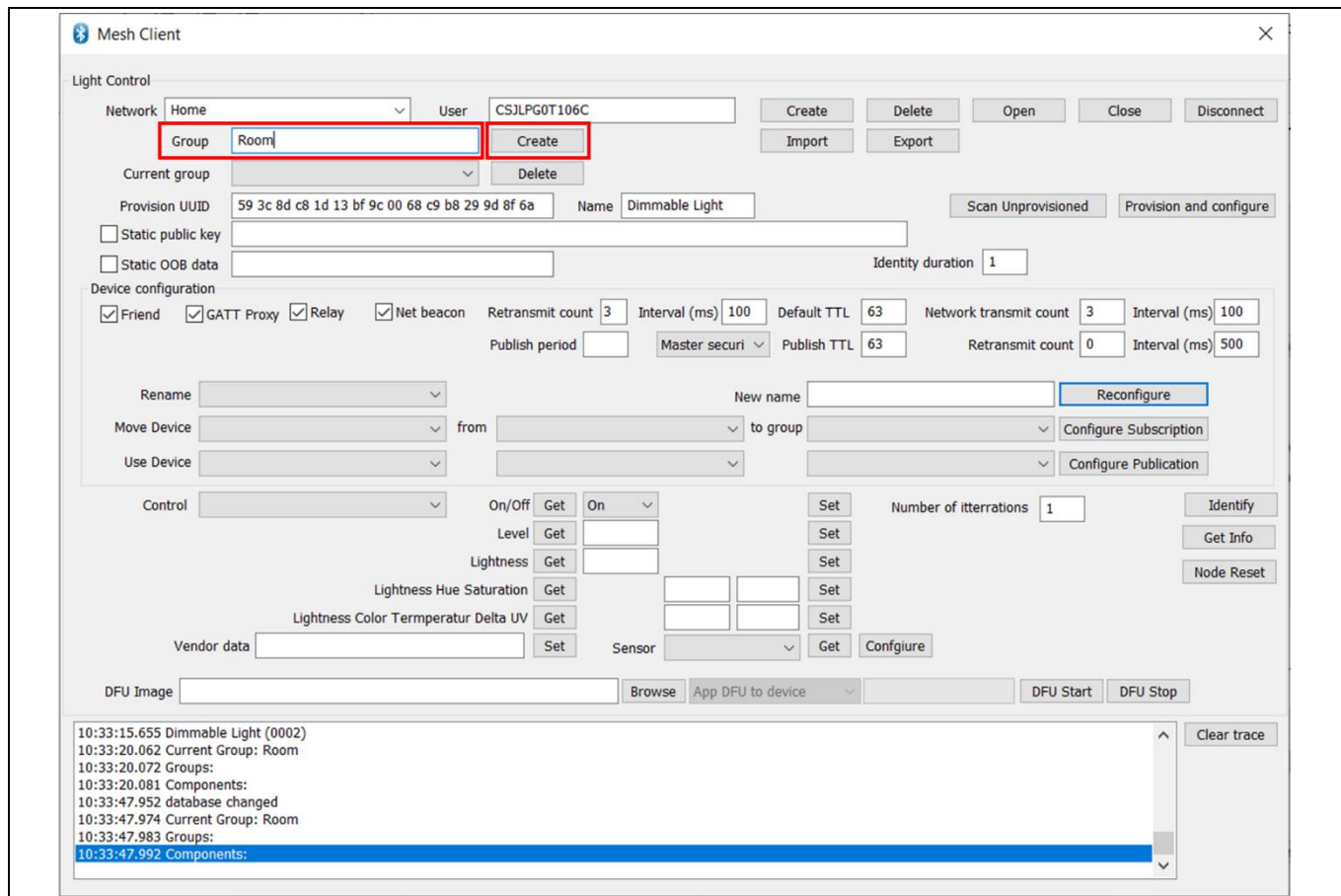


The screenshot displays the Mesh Client application window. The interface is divided into several sections:

- Light Control:** Includes fields for Network (Home), User (CSJLPG0T106C), Group, and Current group. It also features buttons for Create, Delete, Open, Close, Connect, Import, and Export. A red box highlights the Provision UUID (89 71 26 9c c3 60 21 ec 14 76 d7 34 c1 2c 76 df) and the Name (Dimmer).
- Static public key:** A checkbox and a text field.
- Static OOB data:** A checkbox and a text field.
- Identity duration:** A text field set to 1.
- Device configuration:** Includes checkboxes for Friend, GATT Proxy, Relay, and Net beacon. It also has fields for Retransmit count (3), Interval (ms) (100), Default TTL (63), Network transmit count (3), Interval (ms) (100), Publish period, Master securi, Publish TTL (63), Retransmit count (0), and Interval (ms) (500).
- Rename:** A dropdown menu and a text field for the New name.
- Move Device:** A dropdown menu and a text field for the group.
- Use Device:** A dropdown menu and a text field.
- Control:** A dropdown menu and a text field for the On/Off.
- Level:** A text field.
- Lightness:** A text field.
- Lightness Hue Saturation:** A text field.
- Lightness Color Temperatur Delta UV:** A text field.
- Vendor data:** A text field.
- Sensor:** A dropdown menu and a text field.
- DFU Image:** A text field and a Browse button.
- App DFU to device:** A dropdown menu.
- DFU Start:** A button.
- DFU Stop:** A button.
- Log window:** A text area at the bottom showing a log of events. A red box highlights the first three lines: "11:45:23.004 Name:Dimmer (0002)", "11:45:23.091 Can control: LEVEL,", and "11:45:23.125 Current Group: Home".

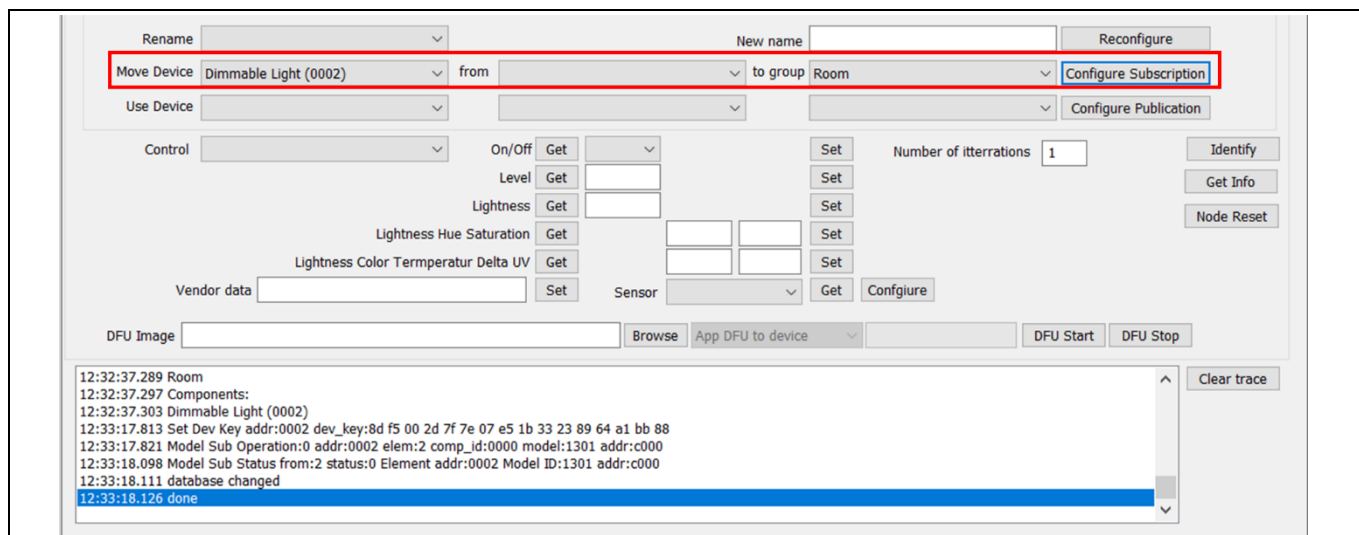
4.2 Creating and Managing Groups

A group can be used to issue commands to several devices at the same time. To create a group in the current network, type in a string in the **Group** field, and click **Create** next to it.



You can then select the group in the Current group field. The nodes now will be provisioned in to the created group.

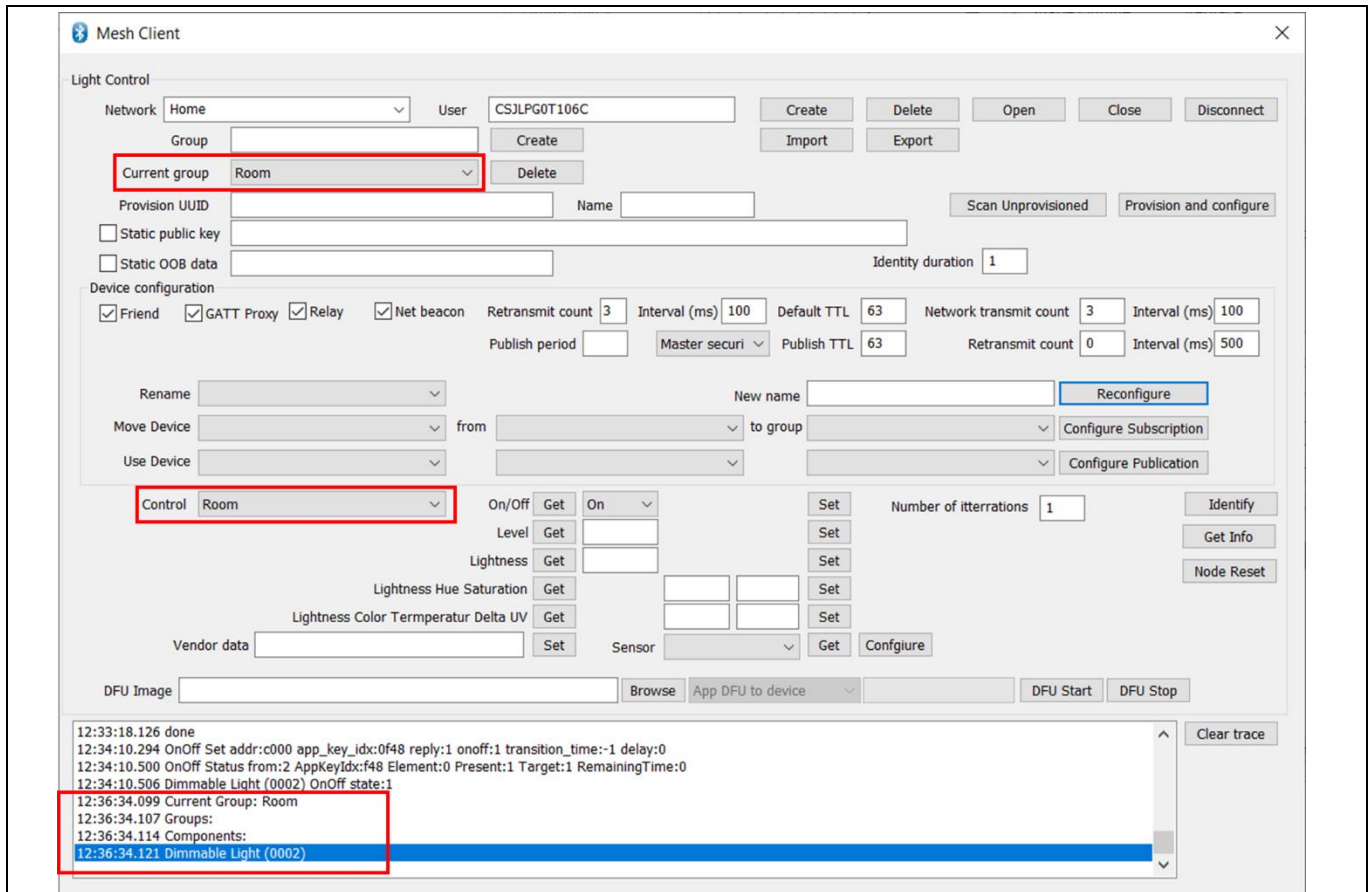
You can also move devices between previously created groups. **Configure Subscriptions** puts the device in to a requested group, the device is now subscribed to process unicast messages destined to that device as well as messages addressed to the group.



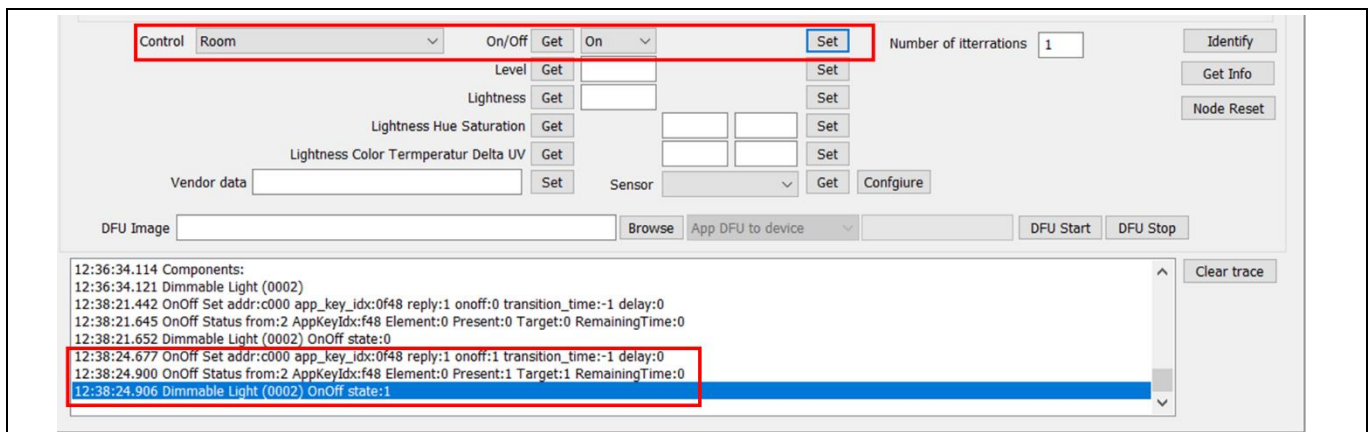
Using the MeshClient Application

If a new device is provisioned while the current selected group is 'Room', the device will be assigned to this group. You do not need to perform the Move operation. In the Control field, you can select an individual device, a group address, or the name of the network to unicast, multicast, or broadcast mesh control messages respectively.

When the current group is changed, the trace window will display the content of the new current group.

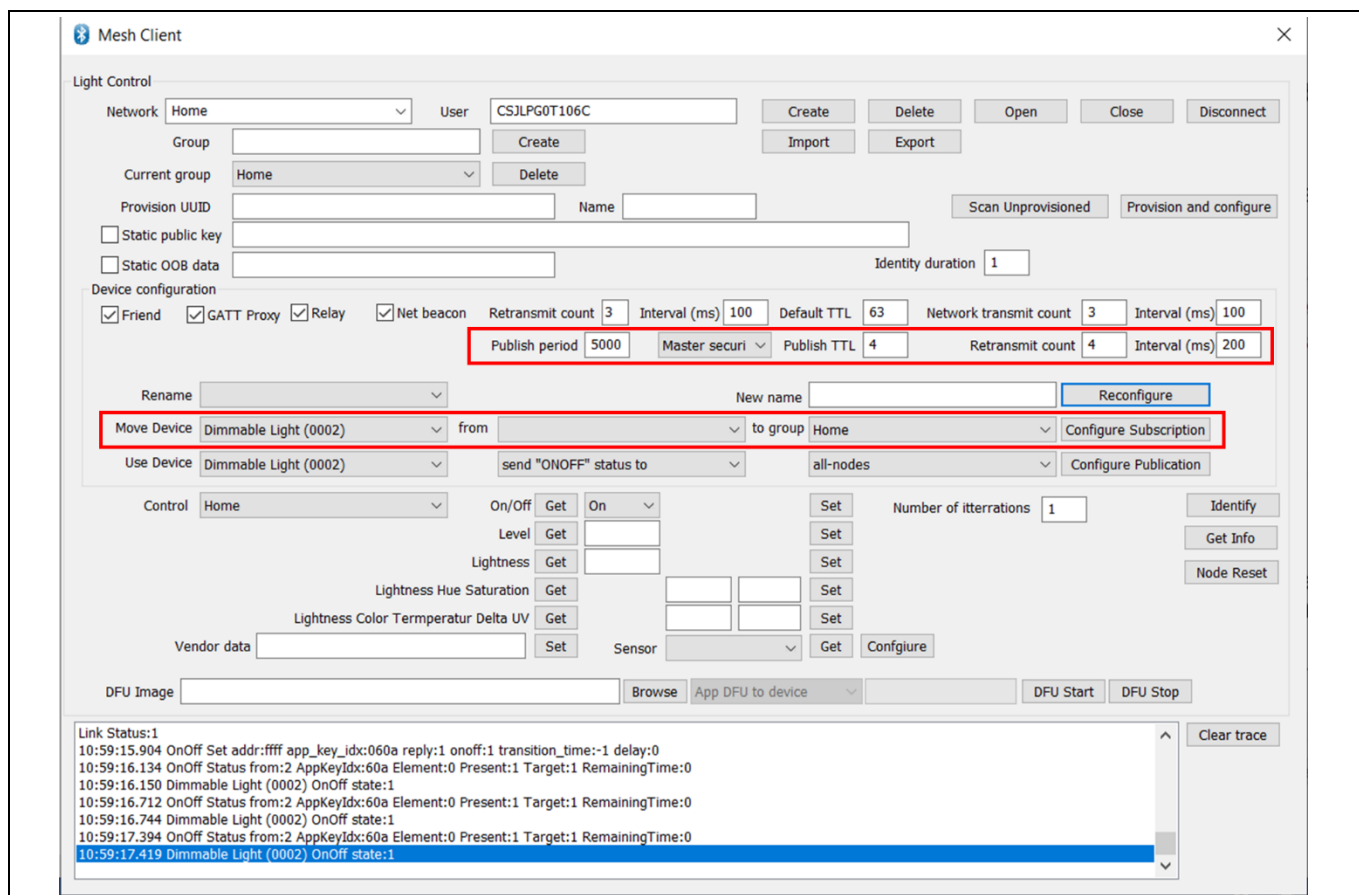


Entire group can be controlled using a single command. Select the group name to be controlled in **Control** field, select the action, and click **Set**.



4.3 Configuring Devices

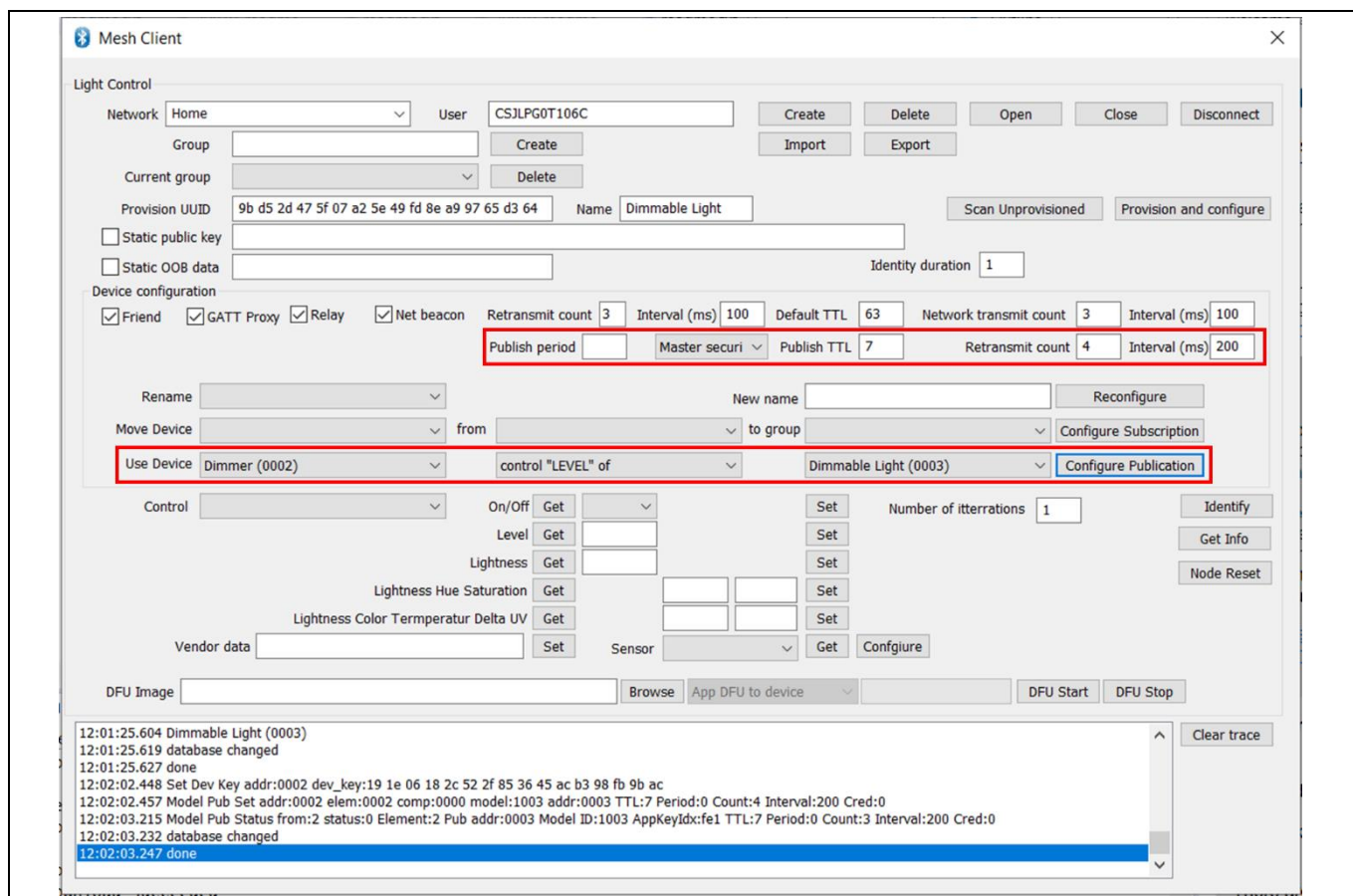
The device configuration section allows you to configure multiple parameters for sending or relaying messages. By default, parameters from the initial provisioning are used; you can specify a new name for an already configured device, select the device in the **Rename** field, and click **Reconfigure**.



A device can also be configured to publish status change to a specific node, and specific group, or to all devices in the network. For example, a light bulb can publish hue/saturation/lightness state periodically.

Similarly, a controlling device can be configured to send messages to individual devices or to a group. This allows you to configure a switch to control one or more lights.

To configure a destination for the messages originated by the device, select the device in the **Use Device** field, select the method (for example, if this is a Dimmer, a LEVEL methods will be available), desired destination, and click **Configure Publication**.



If the Dimmer was provisioned while the group room was selected as the current group, the Dimmer will already be configured to send level to all devices in the group room. However, it can be reconfigured to send messages to any single device, or to any other group in the network.

The publication parameters are used from the Publish period, Publish TTL, Retransmit count, and Interval fields.

4.4 Over-The-Air Device Firmware Upgrade

The device firmware is updated from the build PC to the development kit using the MeshClient DFU interface. Open the mesh **Network**, named **home** in [Figure 3](#). Scan unprovisioned devices, then Provision and Configure. Then, click **Disconnect** (Connect toggles as shown in [Figure 3](#)). Next, select the device from the **Control** dropdown list and click **Connect**. Use **Browse** to select the update image and the image information file. Select **App DFU to device** from the transfer type dropdown list. Click **DFU Start** to begin the transfer.

The image information file is a plain text file formatted as follows:

```
# Company ID (2 bytes)
CID=0x0131

# Firmware ID (2 bytes Product ID + 2 bytes HW Version ID + 4 bytes FW
Version)
FWID=0x3026000101010002
```

MeshClient and ClientControlMesh App User Guide

ModusToolbox

Using the MeshClient Application

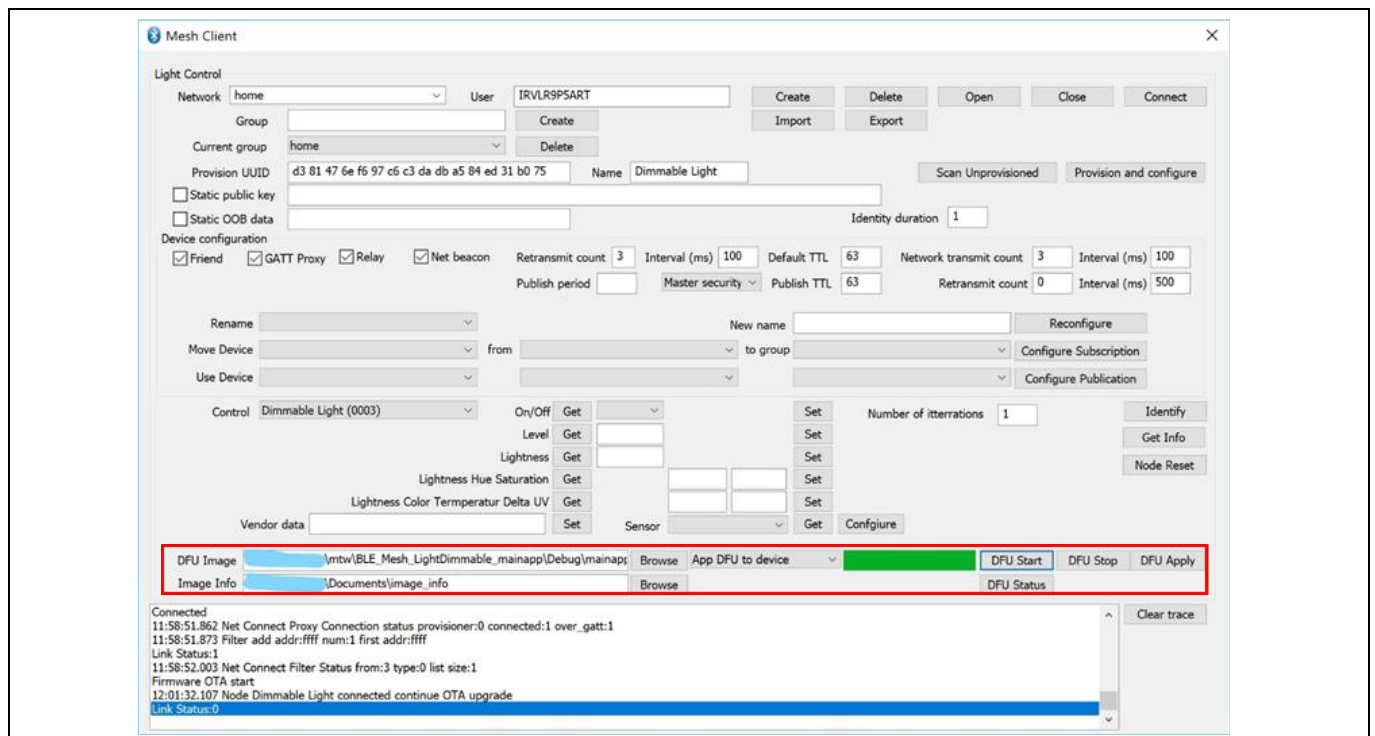


Figure 3 Mesh Client DFU

5 Mesh Performance Testing

5.1 Overview

The common performance indicators that need to be tested to make the assertion of a satisfactory performance in a Mesh network include the ability to:

1. Provision/configure all nodes in the network from a provisioner
2. Perform Mesh model operations like get and set state values on all nodes from a provisioner or the controlling node. For example, Get/Set ON/OFF state and so on.
 - a. When turn ON/OFF lights is issued to all bulbs in the network, how many of the nodes turn ON/OFF?
 - b. When vendor-specific data is send to a destination node, how many of the 50 iterations does the send succeed?
3. Perform FW upgrade on all nodes from a distributor node.

5.2 Key Performance Indicators

Table 1 provides a partial list of key performance indicators/metrics and variables identified to affect the performance of a Bluetooth Mesh Network. For the discussion, the key performance indicators identified for the Mesh Performance are reliability, latency, power consumption, and network throughput. The variables identified to affect the performance indicators are the Number of hops (Time-To-Live), ADV Tx Power, Network transmission and retransmission counts, packet or payload size, and the number of nodes in the network. Each variable is either directly or inversely proportional to the performance indicators as summarized in **Table 1**. For example, as the ADV Tx Power increases, the reliability, power consumption, and the network throughput increase and the latency in the mesh network decreases.

Table 1 Key Performance Indicators and Variables

Performance Indicators	Variables				
	Hops (TTL)	ADV Tx Power	Network / Relay Retransmit count	Packet Length	Number of Nodes
Reliability	Directly	Directly	Directly	Inversely	Directly
Latency	Directly	Inversely	Directly	Directly	Inversely
Power consumption	Directly	Directly	Directly	Directly	Directly
Network throughput	Inversely	Directly	Inversely	Directly	

5.2.1 Probability/Reliability

In a Bluetooth Mesh network, a provisioner node needs to be able to provision, configure and control all nodes in the network and do so reliably. The reliability of the mesh network depends on the reliability with which each of the nodes are accessible and can communicate with each other on-a-need basis. To measure the probability or reliability of performing mesh operations, several common mesh operations are repeated in an iterative manner and the number of success and failure cases are calculated for each operation. Common mesh operations that are used to evaluate the reliability are the ability to provision, configure, and reset a set of nodes, the ability to control the nodes by performing get and set operations on its control states, and the ability to perform DFU updates of nodes in the network.

Mesh Performance Testing

Mesh Performance Testing feature allows you to specify the number of iterations to be performed for a Set Vendor Data operation to evaluate the reliability of the mesh operation. Other operations can be carried out manually from the **Light Control** tab of the application.

5.2.2 Latency

Latency is the measure of the time it takes for a mesh operation to complete. Latency is measured by performing a test mesh operation and evaluating the time it takes from the beginning to the completion of the operation.

Measuring latency in a mesh network provides us with metrics to evaluate the robustness and responsiveness of the nodes in the mesh network. Lower the latency, the better the responsiveness of the network and vice-versa.

Mesh operations usually begin on the source node and end on the destination node. In this scenario, to measure the latency or the time taken for a mesh operation to complete will require that the times on both the source and destination nodes are synchronized at the lowest possible granularity. Synchronization of clocks on both the source and destination nodes at the millisecond granularity may not be easy to setup in the general sense.

An alternative approach to measure the latency considers the round-trip time it takes for a mesh message to reach the destination and acknowledge to be received back at the source. This provides an average latency for one-way transmission.

Latency is affected by the operating parameters of the node like Network transmission count, Network transmission interval, retransmission count, and re-transmission interval. As the number of intermediate relay nodes increases in a network, the latency also increases.

5.2.3 Number of Hops/Time-To-Live (TTL)

Time-To-Live (TTL) value is part of every mesh packet that is transmitted and indicates how each Mesh node that sees the packet should handle the packet. An intermediate node that sees a packet not destined for itself retransmits the packet after it decrements the TTL value present in the packet by one. Each such retransmission can be considered as one hop and the number of hops it takes for a message to reach the destination can help determine an optimal the mesh network layout and power consumption characteristics of the network.

To measure the number of hops it takes a message to reach the destination is based on the TTL value. When an intermediary node retransmits a message, the TTL value for that message is decremented by one.

Varying the TTL does not influence the number of hops it will take for a mesh message to reach the destination. However, estimating the number of hops it takes for a message to reach the destination may provide insights into how to better adjust the layout of nodes in the mesh network, so that the message reaches the destination is least number of hops and thus decreasing the power consumption, latency and improving network throughput. TTL value is configured on the **Light Control** tab of the application.

For networks with large number of relay nodes, the number of hops, latency, and power consumption also increase, and therefore selecting the number of relay nodes becomes critical to the network performance.

5.2.4 Power Consumption

Power consumption at each node is affected by several operating parameters of the node. One of the parameter that directly affects the power consumption is Advertisement Transmission Power (ADV Tx Power). This value defines the power setting used by the node when the node transmits or retransmits advertisements

to communicate with other nodes. Parameter to set the ADV Tx power is configurable in the mesh core for each Mesh application.

Currently, Tx Power setting can be configured in 20819A1 chips using the following setting in the code. The range of values available for Tx Power setting is between [0-4], where 0 (MULTI_ADV_TX_POWER_MIN) indicates the lowest Tx power setting and 4 (MULTI_ADV_TX_POWER_MAX) indicates the maximum Tx power setting supported by the chip. This value is configurable from the **Mesh Performance Testing** tab for each of the nodes provisioned and configured.

```
uint8_t wiced_bt_mesh_core_adv_tx_power = MULTI_ADV_TX_POWER_MAX;
```

Using Mesh Performance feature, it is possible to adjust the ADV Tx Power of a node in the network to make sure that the node and the network are performing optimally. Few common reasons that many necessitate adjusting the ADV Tx Power of an individual node are:

- Number of network nodes in the proximity
 - Too few nodes in the proximity may need higher Tx Power
 - Too many nodes may suggest lowering Tx Power
- Interference from surroundings like walls, structures, and so on
 - Obscure or heavily hindered locations may require a node to have higher Tx Power to function effectively
 - Lower Tx Power setting may be necessary in situations where range of the node needs to be restricted to smaller area.

Optimally configuring the ADV Tx Power at each node in the Mesh Network based on the requirements may improve mesh performance metrics as against to configuring all nodes to the same ADV Tx Power settings.

Mesh Performance testing feature allows you to configure the ADV Tx Power for both local and the remote mesh nodes from the user interface.

5.2.5 Packet Length/Payload Size

In Bluetooth Mesh, the payloads above 12 bytes are segmented and reassembled at the source and the destination, respectively. Each operation of segmentation, network transmission, and reassembly increases the latency of the mesh operation. As the payload size increases the latency also increases, therefore payload size is a key design parameter that must be considered when designing the mesh application and message structures. As the payload size increases, the transmission of segmented packets increases, which further increases the power consumption of the network.

MeshPerformance feature allows you to indicate the payload length to be used for the testing procedure.

5.2.6 Network/Relay Retransmission

Mesh nodes, both the transmitter and the relay nodes, are configured to repeat the same message multiple times at the network level to improve the reliability by compensating for the packet loss over the air due to interference. The number of repetitions and the interval of repetition is usually set at the time of configuration of the mesh node. However, it might be advantageous to reconfigure and fine-tune the number of retransmissions and the interval after the network is setup to improve the mesh performance indicators, namely the latency, number of hops, power consumption, and the network throughput.

MeshPerformance feature allows you to disable and enable network retransmission for the vendor data set operation specifically to study the effect of network retransmissions on the latency. Network retransmission can be disabled or enabled for both local and the remote devices by selecting the device in the dropdown on the **Mesh Performance** tab. Network transmission count and interval can be configured from the **Light Control** tab of the application.

5.2.7 Methods to Measure Performance Indicators

Evaluation of network-level mesh performance indicators will require evaluation of performance indicators for operations between nodes of the network. There are two ways to evaluate the performance indicators: a direct/one-way approach and an indirect/round-trip approach.

5.2.8 Direct/One-way Approach

A direct or one-way approach would require that all nodes in the mesh network operate on a synchronized network time and each node logs the information precisely when mesh commands and events are handled. Subsequently, logs from all nodes are merged and synchronized for analysis and a complete picture of the flow of mesh messages across the mesh network can be visualized.

This consolidated log information from all nodes can help evaluate the success and failure cases for reliability, latency measurements for mesh operations, number of hops taken by a message to arrive at the destination, and to study the optimal TTL it will take for the mesh operation to complete.

The following components are required to implement this approach:

- Mesh nodes – 20819A1 boards configured with mesh applications
- MeshClientControl.sln (MCC) - Application running on a PC that will connect to and act as an MCU attached to the mesh node. MCC application will collect and log mesh application state events from the mesh node to which it is connected. Multiple instances of MCC applications may be run on a single PC to allow collection of logs from as many mesh nodes connected to that PC. Alternatively, a single PC running a single instance of MCC may be setup to connect to one mesh node.

To collect and synchronize the log data collected from all MCC applications into a central location, and to avoid manual operations of copying and synchronizing the log data after each iteration of the testing procedure, an implementation based on a UDP Server and multiple UDP Clients is recommended.

A listening UDP message server would be implemented as part of the MCC attached to the Mesh Controlling node/provisioner/Client, and UDP message clients would be implemented in the MCCs attached to the Mesh nodes acting as the Mesh model server nodes.

The following are the steps to setup a mesh node framework for this approach:

1. Modify the MeshClientControl.sln application to include UDP Server and Clients.
2. Attach the MCC with the UDP Server to the 20189-board running the Mesh Application Client (For example: ON/OFF client, vendor client, and so on).
3. Attach MCC with UDP clients to the 20819 boards running Mesh Application Servers (For example: ON/OFF Server, Vendor server).
4. When the MCC application attached to the Mesh Application Servers receive events from the board, they will forward the same to the UDP Server, which will collect all incoming logs along with the IP address of the machine they are originating from and the timestamp. At this point, the logs from one MeshApplication Client and the MeshApplication Servers are consolidated, as shown in Figure 4.

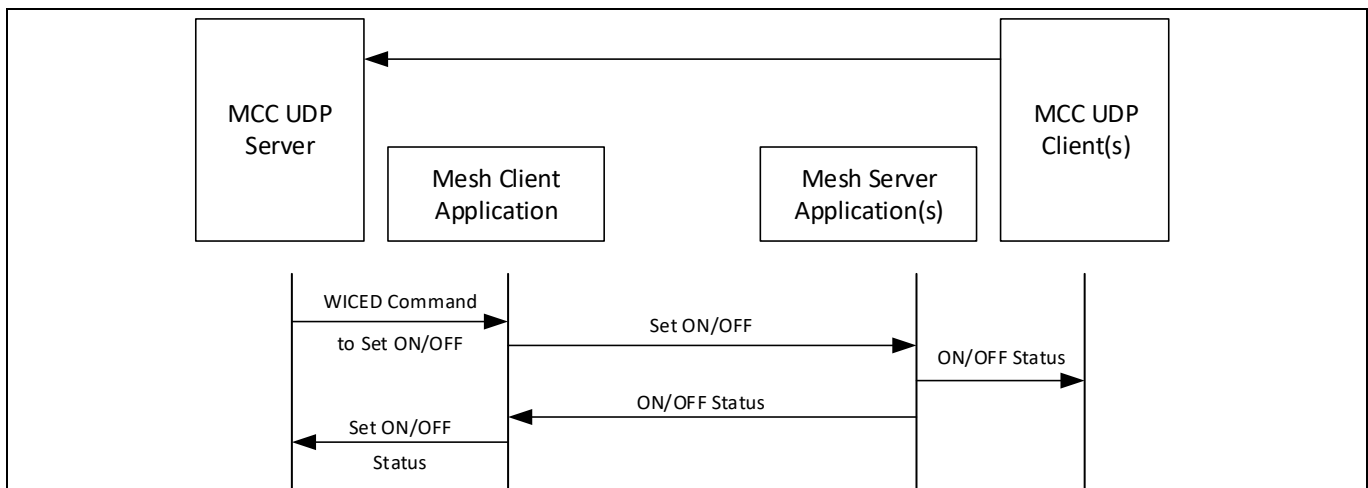


Figure 4 Direct/One-way Approach

Based on the analysis of requirements of the approach, the following implementation-related issues were identified:

- Need to synchronize time across all nodes of the mesh network at milli-second accuracy
- Ability to account for latency of logging information using UDP protocol over the local WiFi network
- Additional overhead of attaching a PC to each of the mesh nodes to facilitate collection of logs and transmission

5.2.9 Indirect/Round-trip Approach

In view of many the implementation issues, an alternative round-trip close approximation approach to evaluate the latency and TTL metrics is proposed.

To measure the latency, a mesh message would be sent from a source node to a destination node and the source node would wait to receive a response from the destination node to confirm that the mesh operation completed successfully. Absence of a response from the destination message in a reasonable period would be considered a failure. Period to register a failure of a given operation may depend on the timeout values. The total time between the initiation of the operation and the completion/failure events can be assumed to be the total round-trip time and half of that time would give the required latency of messages between any two nodes.

To measure number of hops based on the change in the TTL values, a customized MeshVendorClient/Server can be used. MeshVendorClient will send a message to custom VendorServer implementation in the mesh_perf_testing application with a certain TTL value as set in the node configuration.

- Packets traverses across the mesh network and reach the VendorServer.
- When the MeshVendorServer receives the message from the VendorClient, it can recognize the TTL value in the received packet and echo the value back to the VendorClient as a vendor message. Upon receiving the response back from the VendorServer, VendorClient can obtain the TTL value that the VendorServer received and evaluate the difference between the original TTL value that was sent and the TTL value that the VendorServer received. This will provide the number of hops it took for the message to traverse the mesh network from the VendorClient to the VendorServer as shown in Figure 5.

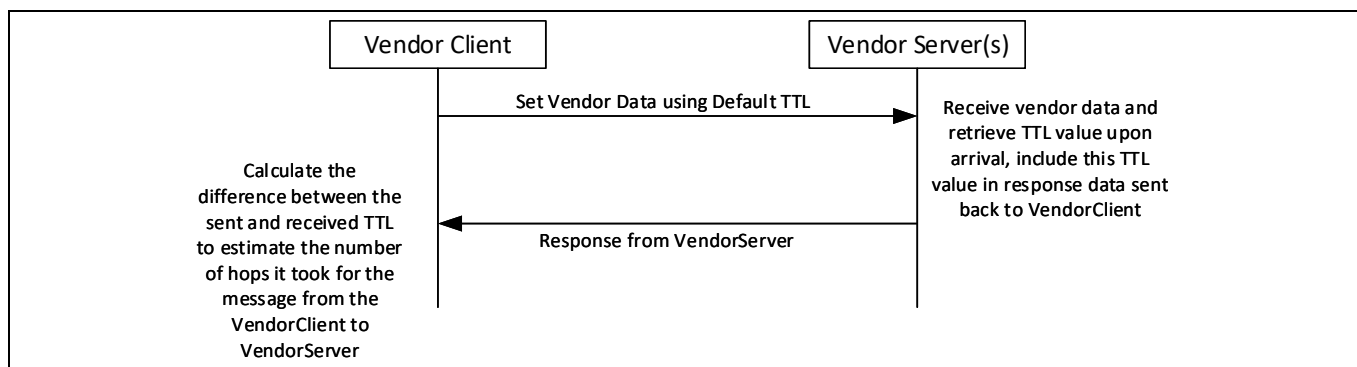


Figure 5 Indirect/Round-trip Approach

5.3 Testing Procedure

Use latest ModusToolbox 2.2+ IDE and any platform that supports Mesh, such as 20819s, 20735, and so on, and install the latest BTSDK. At the time of publication of this document, BTSDK 2.8.0 is available.

- VendorClient (Part of Mesh Provision Client app)
 - Create and build the Mesh Snip Provision Client project for the CYBT-213043-MESH BSP, which has Mesh Vendor Client.
- VendorServer (Part of Mesh Performance Testing app)
 - Create and build the Mesh Snip Performance Testing project for the CYBT-213043-MESH BSP.
- Program one device with VendorClient, and several devices with VendorServer.
- Execute *ClientControlMesh.exe* (available in the latest BTSDK):

mtb_shared\wiced_btstack\tools\btstack-host-apps-mesh\VS_ClientControl\Release\ClientControlMesh.exe

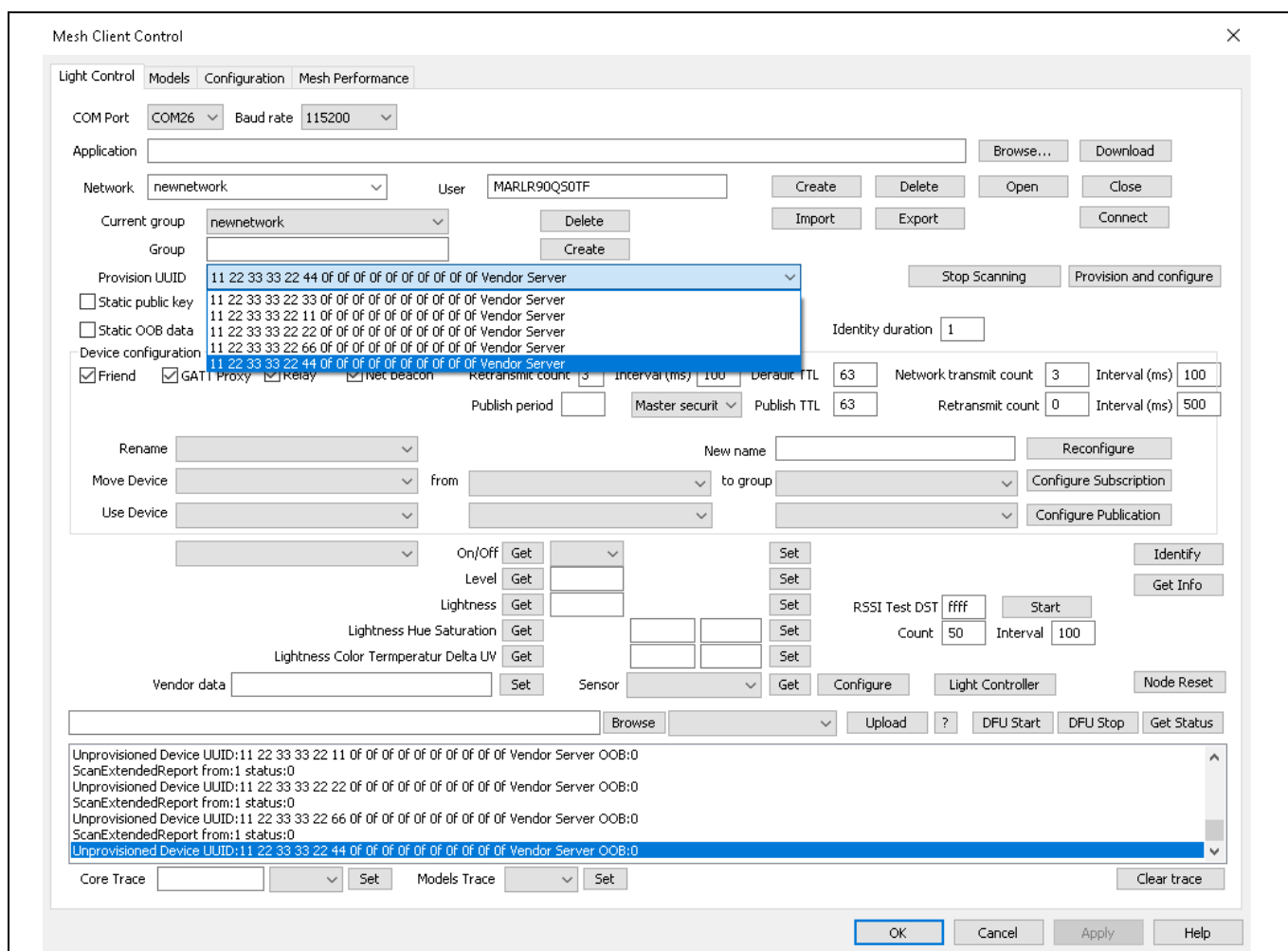
Launch *ClientControlMesh.exe* from the command line as shown below, and open the WICED HCI UART (usually the first COM port) and connect to the VendorClient. To enable tracing to the file, add `-T` to command line:

```
C:\ ClientControlMesh.exe -T
```

All traces from *ClientControlMesh.exe* will be collected in *traces.txt* in the same folder. If possible, connect a PC to each VendorServer board and collect PUART traces. Open TeraTerm and connect the WICED Peripheral UART port (usually COM port with greater value) (921600 baud) to PUART on VendorClient/VendorServer and make sure traces are received.

Follow these steps, on the ClientControlMesh – Light Control tab, to test reliability of sending and receiving data between VendorClient and VendorServer:

1. Enter a **Network** name and create the network.
2. Open the network.
3. Click **Scan Unprovisioned** and scan for unprovisioned devices.
4. Click **Stop Scanning** once the device is found. This step is an essential step.



- Click **Provision and Configure** to provision the device. Let the provisioning complete. Provision all nodes that need to be added to the mesh network
- Provision multiple VendorServers on the **Light Control** tab. All provisioned devices will show in the devices list. For instance, (0002) and so on.

Mesh Performance Testing

Upon provisioning all devices, place the devices in the mesh network topology to be used for the testing.
See [Figure 5](#).

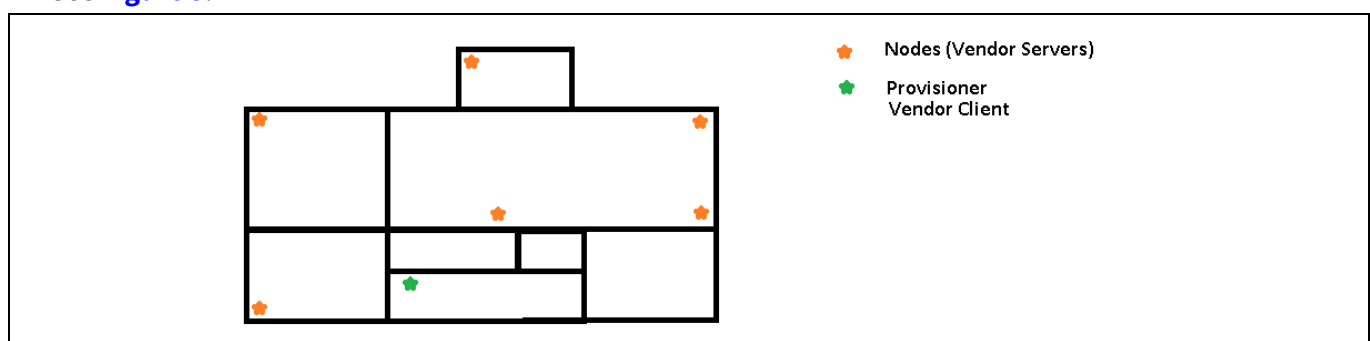


Figure 5 Devices in Mesh Network Topology

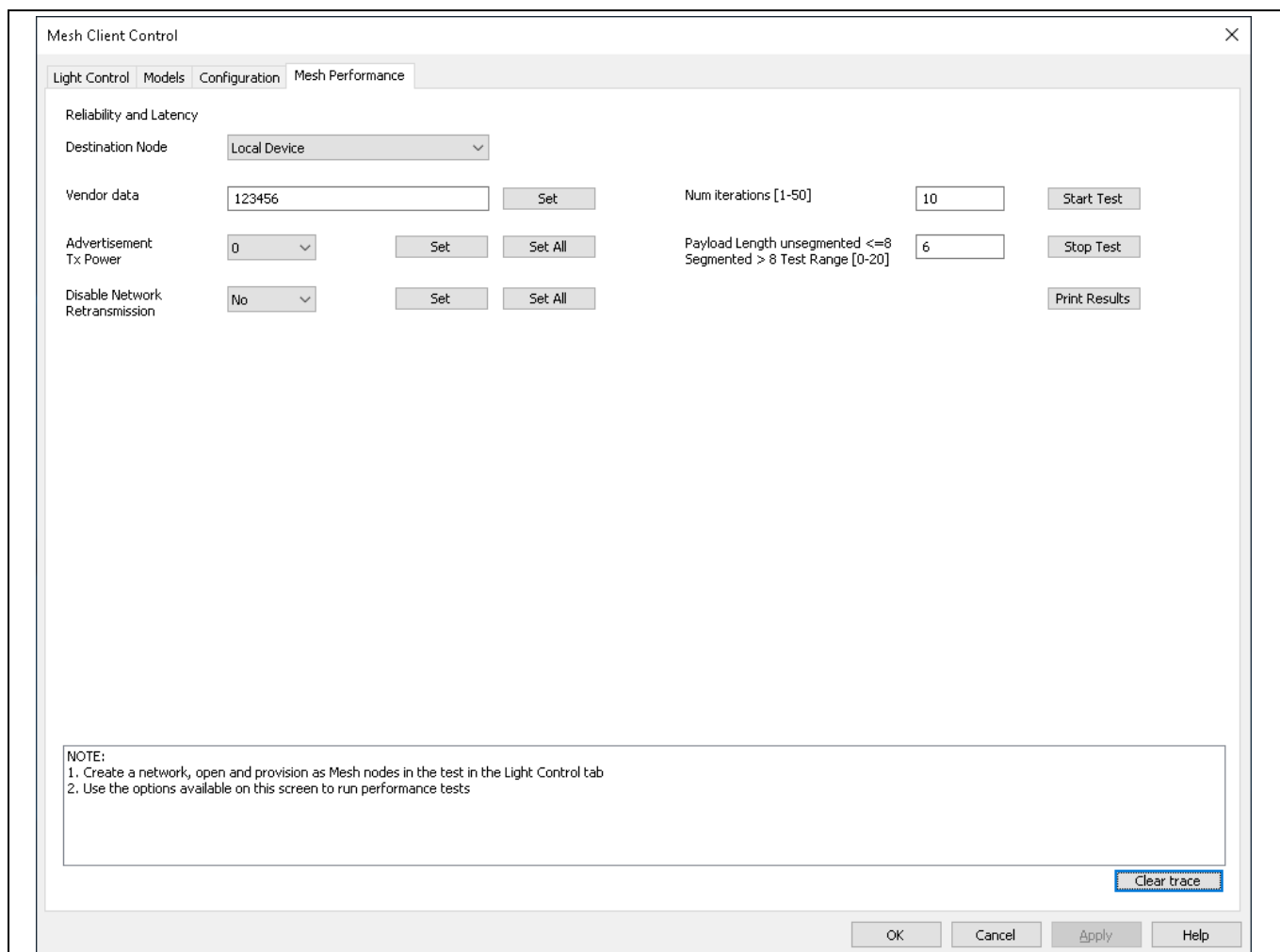
6. Select the destination device from the **Destination Node** drop-down list.
7. Enter **123456** in **Vendor data** and click **Set**.

Traces should indicate the data is sent to VendorServer and data echoed back by the VendorServer and received at VendorClient. This verifies that the mesh data can be sent to the destination node.

8. Switch to the **Mesh Performance** tab to perform a variety of tests.

Mesh Performance Testing

Mesh Performance tool can be used to evaluate the effect of ADV Tx Power, Packet Length, and Network retransmission. Vendor Data Set operations are used to configure values on the mesh nodes. Default TTL and Relay/Network retransmission count, and the interval can be configured using the **Light Control** tab at the device level (then applicable for all mesh operations) before or after provisioning the device. Currently, Mesh Performance tool does not support evaluation of Network Throughput and Power Consumption.



The screenshot shows the 'Mesh Client Control' application window with the 'Mesh Performance' tab selected. The window has a title bar with a close button (X). Below the title bar are four tabs: 'Light Control', 'Models', 'Configuration', and 'Mesh Performance'. The 'Mesh Performance' tab is active and contains the following controls:

- Reliability and Latency** section:
 - 'Destination Node' dropdown menu showing 'Local Device'.
- Vendor data** section:
 - Input field with '123456' and a 'Set' button.
 - 'Num iterations [1-50]' input field with '10' and a 'Start Test' button.
- Advertisement Tx Power** section:
 - Dropdown menu with '0' and 'Set' and 'Set All' buttons.
 - 'Payload Length unsegmented <=8' input field with '6' and a 'Stop Test' button.
 - 'Segmented > 8 Test Range [0-20]' input field with '6'.
- Disable Network Retransmission** section:
 - Dropdown menu with 'No' and 'Set' and 'Set All' buttons.
 - 'Print Results' button.

At the bottom of the window, there is a 'NOTE' box with the following text:

1. Create a network, open and provision as Mesh nodes in the test in the Light Control tab
2. Use the options available on this screen to run performance tests

Below the note box is a 'Clear trace' button. At the very bottom of the window are four buttons: 'OK', 'Cancel', 'Apply', and 'Help'.

On the **Mesh Performance** tab, you can perform tests to gather and understand data on the reliability, latency, number of hops, payload size as they relate to the Mesh performance. The destination nodes list displays the nodes that were previously provisioned along with the local device.

Local Device only supports the operations to set local ADV Tx Power and Disable/Enable Network retransmission.

Set operations are intended to work once for the device selected in the destination nodes list. **Set All** operations can be used to set the ADV Tx Power and Disable Network retransmission operations for all destination mesh nodes and the local device.

ADV Tx Power values range from 0 - 4. The value 0 indicates least power and 4 indicates maximum power. Set the value to 0 if the mesh network is dense (lots of devices in a small space) and make sure that you get at least a few hops.

Mesh Performance Testing

Testing message reliability (one time)

9. Select the destination device from the **Destination Node** dropdown list. Enter **123456** in **Vendor data** and click **Set**.

Testing message reliability (multiple times in a loop)

10. Select the destination device from the **Destination Node** dropdown list.
Specify the number of iterations and the payload size, in number of bytes, to be used for this test. Click **Start Test**. This will trigger a periodic timer (4 second interval) to send data to the destination device.
In this test, data is sent N (as specified above) times to the destination and you expect to receive the data echoed back N times.
The default test runs for 10 times with a payload size of 3 bytes.

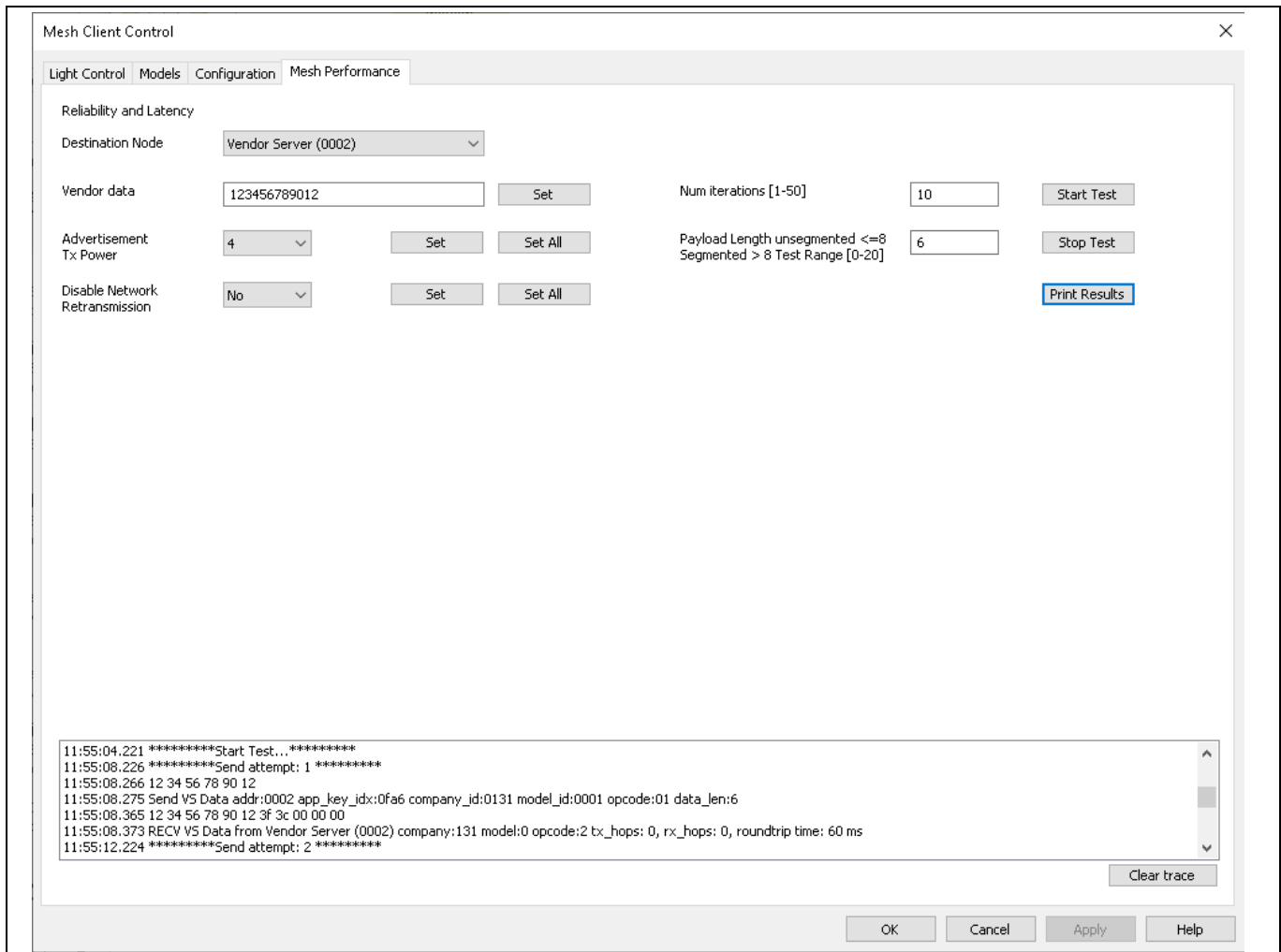
Testing reliability with relay nodes in between (multiple nodes)

11. Select a destination device that is farther from the provisioner node such that there are other VendorServers between the VendorClient and the destination VendorServer and the data sent must be relayed or must hop over multiple VendorServers to reach the destination device.
Repeat the test with the selected node using both the one-time test and iteration test.
Traces indicate the number of hops (Tx hops) it takes for the message to reach the destination node. Rx hops indicates the number of hops the message took on its way back to the source device, and the roundtrip time it took for the message to travel back and forth.

MeshClient and ClientControlMesh App User Guide

ModusToolbox

Mesh Performance Testing



Mesh Client Control

Light Control | Models | Configuration | Mesh Performance

Reliability and Latency

Destination Node: Vendor Server (0002)

Vendor data: 123456789012 [Set]

Num iterations [1-50]: 10 [Start Test]

Advertisement Tx Power: 4 [Set] [Set All]

Payload Length unsegmented <=8: 6 [Stop Test]

Disable Network Retransmission: No [Set] [Set All]

[Print Results]

Log Window:

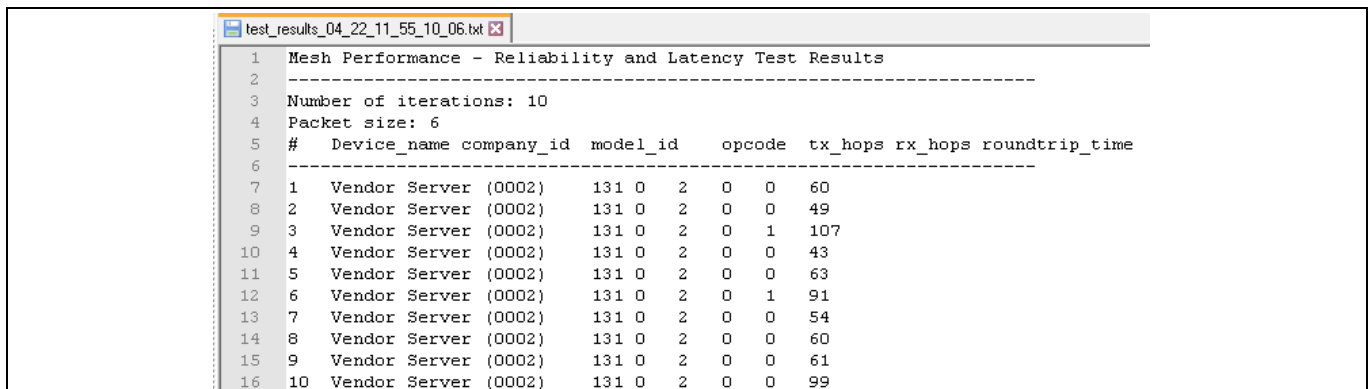
```

11:55:04.221 *****Start Test...*****
11:55:08.226 *****Send attempt: 1*****
11:55:08.266 12 34 56 78 90 12
11:55:08.275 Send V5 Data addr:0002 app_key_idx:0fa6 company_id:0131 model_id:0001 opcode:01 data_len:6
11:55:08.365 12 34 56 78 90 12 3f 3c 00 00 00
11:55:08.373 RECV V5 Data from Vendor Server (0002) company:131 model:0 opcode:2 tx_hops: 0, rx_hops: 0, roundtrip time: 60 ms
11:55:12.224 *****Send attempt: 2*****
  
```

[Clear trace]

[OK] [Cancel] [Apply] [Help]

12. Click **Print Results** to print the results of the multiple loop test in to a tab-delimited file, which can be used for data analysis.



test_results_04_22_11_55_10_06.txt

Mesh Performance - Reliability and Latency Test Results							

Number of iterations: 10							
Packet size: 6							
#	Device_name	company_id	model_id	opcode	tx_hops	rx_hops	roundtrip_time

1	Vendor Server (0002)	131 0	2	0	0	60	
2	Vendor Server (0002)	131 0	2	0	0	49	
3	Vendor Server (0002)	131 0	2	0	1	107	
4	Vendor Server (0002)	131 0	2	0	0	43	
5	Vendor Server (0002)	131 0	2	0	0	63	
6	Vendor Server (0002)	131 0	2	0	1	91	
7	Vendor Server (0002)	131 0	2	0	0	54	
8	Vendor Server (0002)	131 0	2	0	0	60	
9	Vendor Server (0002)	131 0	2	0	0	61	
10	Vendor Server (0002)	131 0	2	0	0	99	

#	Device_name	company_id	model_id	opcode	tx_hops	rx_hops	undrtrip_time		#	Device_name	company_id	model_id	opcode	tx_hops	rx_hops	undrtrip_time		#	Device_name	mpany_id	model_id	opcode	tx_hops	rx_hops	trip_time
1	Vendor Ser	131	0	2	1	1	136		1	Vendor Server	131	0	2	0	1	75		1	Vendor Serve	131	0	2	0	0	44
2	Vendor Ser	131	0	2	1	2	180		2	Vendor Server	131	0	2	1	0	74		2	Vendor Serve	131	0	2	0	0	49
3	Vendor Ser	131	0	2	2	1	244		3	Vendor Server	131	0	2	1	0	67		3	Vendor Serve	131	0	2	0	0	65
4	Vendor Ser	131	0	2	1	1	122		4	Vendor Server	131	0	2	0	1	103		4	Vendor Serve	131	0	2	1	2	193
5	Vendor Ser	131	0	2	1	1	134		5	Vendor Server	131	0	2	0	1	71		5	Vendor Serve	131	0	2	0	0	59
6	Vendor Ser	131	0	2	1	3	237		6	Vendor Server	131	0	2	0	0	38		6	Vendor Serve	131	0	2	0	0	50
7	Vendor Ser	131	0	2	2	2	246		7	Vendor Server	131	0	2	0	0	39		7	Vendor Serve	131	0	2	1	1	210
8	Vendor Ser	131	0	2	1	1	144		8	Vendor Server	131	0	2	0	0	59		8	Vendor Serve	131	0	2	0	0	38
9	Vendor Ser	131	0	2	2	2	201		9	Vendor Server	131	0	2	0	1	82		9	Vendor Serve	131	0	2	1	0	86
10	Vendor Ser	131	0	2	1	1	112		10	Vendor Server	131	0	2	0	0	56		10	Vendor Serve	131	0	2	0	1	88
					Tx Power Network Retransmissions	Enabled	0						Tx Power Network Retransmissions	Enabled	4							Tx Power Network Retransmissions	Disabled	4	

The analysis of the following relationships is key to evaluating and optimizing the performance in a mesh network:

- Latency versus Payload Size
- Number of Hops versus ADV Tx Power
- Latency versus Number of Hops
- Latency versus ADV Tx Power
- Latency versus Number of Nodes (when nodes are added/removed to/from an existing network)

The following changes need to be made to the applications to configure for debug traces

- Changes to the provision_client and mesh_perf_testing_app to enable debug traces
- Changes to Makefiles to enable debug traces

Note: Enabling debug traces may affect the performance results. However, traces may be required to identify and report issues.

```
mesh_app_init(..) function
#if 1
// Set Debug trace level for mesh_models_lib and mesh_provisioner_lib
wiced_bt_mesh_models_set_trace_level (WICED_BT_MESH_CORE_TRACE_INFO);
#endif

#if 1
// Set Debug trace level for all modules but Info level for CORE_AES_CCM module
wiced_bt_mesh_core_set_trace_level(WICED_BT_MESH_CORE_TRACE_FID_ALL,
WICED_BT_MESH_CORE_TRACE_DEBUG);
wiced_bt_mesh_core_set_trace_level(WICED_BT_MESH_CORE_TRACE_FID_CORE_AES_CCM,
WICED_BT_MESH_CORE_TRACE_INFO);
#endif
```

Code Listing 2 mesh_perf_testing_app.c

```
mesh_app_init(..) function
#ifdef 1
// Set Debug trace level for mesh_models_lib and mesh_provisioner_lib
wiced_bt_mesh_models_set_trace_level (WICED_BT_MESH_CORE_TRACE_INFO);
#endif
#ifdef 1
// Set Debug trace level for all modules but Info level for CORE_AES_CCM module
wiced_bt_mesh_core_set_trace_level(WICED_BT_MESH_CORE_TRACE_FID_ALL,
WICED_BT_MESH_CORE_TRACE_DEBUG);
wiced_bt_mesh_core_set_trace_level(WICED_BT_MESH_CORE_TRACE_FID_CORE_AES_CCM,
WICED_BT_MESH_CORE_TRACE_INFO);
#endif
```

Following are the changes to be made to the Makefiles of both applications:

```
# default target
```

```
TARGET=CYBT-213043-MESH
```

```
# These flags control whether the prebuilt mesh libs (core, models, and
provisioner)
```

```
# will be the trace enabled versions or not
```

```
MESH_MODELS_DEBUG_TRACES ?= 1
```

```
MESH_CORE_DEBUG_TRACES ?= 1
```

```
MESH_PROVISIONER_DEBUG_TRACES ?= 1
```


References

References

- [1] Mesh Profile Specification v1.0
- [2] Mesh Models Specification v1.0
- [3] Mesh Provisioner Database Specification v1.0
- [4] [**AN227069 - Getting Started with Bluetooth Mesh**](#)

Revision history

Document version	Date of release	Description of changes
**	05/01/2018	Initial release
*A	04/24/2019	Removed Associated Part Family Updated for BT SDK release
*B	10/15/2019	Updated for ModusToolbox 2.0
*C	02/11/2020	Updated for latest WICED Studio and ModusToolbox changes
*D	2020-06-10	Added Mesh Performance Testing information

Trademarks

All referenced product or service names and trademarks are the property of their respective owners.

Edition 2020-06-10

Published by

Infineon Technologies AG

81726 Munich, Germany

© 2020 Infineon Technologies AG.

All Rights Reserved.

Do you have a question about this document?

Go to www.cypress.com/support

Document reference

002-26575 Rev. *D

IMPORTANT NOTICE

The information given in this document shall in no event be regarded as a guarantee of conditions or characteristics ("Beschaffenheitsgarantie").

With respect to any examples, hints or any typical values stated herein and/or any information regarding the application of the product, Infineon Technologies hereby disclaims any and all warranties and liabilities of any kind, including without limitation warranties of non-infringement of intellectual property rights of any third party.

In addition, any information given in this document is subject to customer's compliance with its obligations stated in this document and any applicable legal requirements, norms and standards concerning customer's products and any use of the product of Infineon Technologies in customer's applications.

The data contained in this document is exclusively intended for technically trained staff. It is the responsibility of customer's technical departments to evaluate the suitability of the product for the intended application and the completeness of the product information given in this document with respect to such application.

For further information on the product, technology, delivery terms and conditions and prices please contact your nearest Infineon Technologies office (www.infineon.com).

WARNINGS

Due to technical requirements products may contain dangerous substances. For information on the types in question please contact your nearest Infineon Technologies office.

Except as otherwise explicitly approved by Infineon Technologies in a written document signed by authorized representatives of Infineon Technologies, Infineon Technologies' products may not be used in any applications where a failure of the product or any consequences of the use thereof can reasonably be expected to result in personal injury.