# MeshClient and ClientControlMesh app user guide

## ModusToolbox™ software

## About this document

### Scope and purpose

This document provides quick start instructions for the MeshClient and the ClientControlMesh applications, which are part of AIROC™ Bluetooth® SDK (BTSDK) and ModusToolbox™ software.

### Intended audience

This document is intended for embedded application developers using ModusToolbox™ software to implement and test Bluetooth® Mesh-based solutions with AIROC™ Bluetooth® devices.

## Table of contents

**Table of contents**

# 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

The Bluetooth® SDK (BTSDK) in ModusToolbox™ software offers a 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. BTSDK and ModusToolbox™ software support Bluetooth® Mesh on AIROC™ CYW20706, CYW20735, CYW20719, CYW20721, CYW20819, CYW20820, and CYW20835, and Infineon modules such as CYBT-213043-02 based on these silicon devices.

The MeshClient and the ClientControlMesh applications provide a sample implementation that show how to use the interfaces exposed by the AIROC™ Mesh libraries. The MeshClient application 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. This 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 an Infineon device to act as a client. Therefore, this application requires an extra evaluation board to be connected to the PC for Mesh operation. Any of the Infineon devices that support Bluetooth® Mesh can be used for this application irrespective of the device used by Mesh nodes. The ClientControlMesh application can be used with any version of Windows. A version of the ClientControlMesh application is also provided for Linux and macOS.

The MeshClient and the ClientControlMesh Windows applications are installed to the ModusToolbox™ software user workspace when any BTSDK Mesh embedded project is created.

- **App paths in ModusToolbox™ software**

  **MeshClient** and **ClientControlMesh** applications are provided in the *mtb_shared/wiced_btsdk* project in the Eclipse IDE for ModusToolbox™ software, which is created and used by any AIROC™ Bluetooth® application created in the IDE.

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

  *mtb_shared\wiced_btsdk \tools\btsdk-host-peer-apps-mesh\<branch>\peer\Windows\MeshClient\Release\x86*

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

  *mtb_shared\wiced_btsdk\tools\btsdk-host-peer-apps-mesh\<branch>\host\VS_ClientControl*

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

  *mtb_shared/wiced_btsdk/tools/btsdk-host-peer-apps-mesh/<branch>/host/Qt_ClientControl*

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

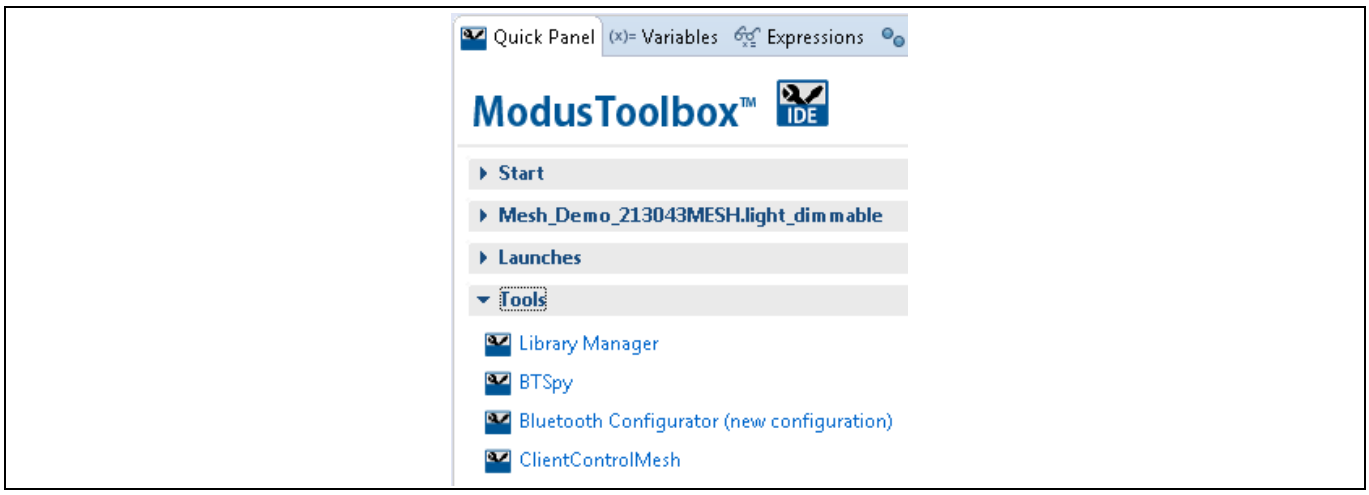



**Figure 1 Tools in Quick Panel**

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

The ClientControlMesh and the MeshClient applications are also supplied as source code and can be built using Microsoft Visual Studio 2019 or later release on Windows. The ClientControlMesh application can also be built on Linux and macOS using Qt Creator 5.0.2 and Qt 5.9.1 for Desktop. Source code and project files for each can be found in the same folder trees where the prebuilt executables are found as discussed above.

### Operating system (OS) requirements

- **MeshClient** - The MeshClient application relies upon the Windows 10 Bluetooth® stack version.
- **ClientControlMesh** - The ClientControlMesh application does not use the Windows Bluetooth® stack and can be executed on any version of Windows, Linux, and macOS supported by ModusToolbox™ software.

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

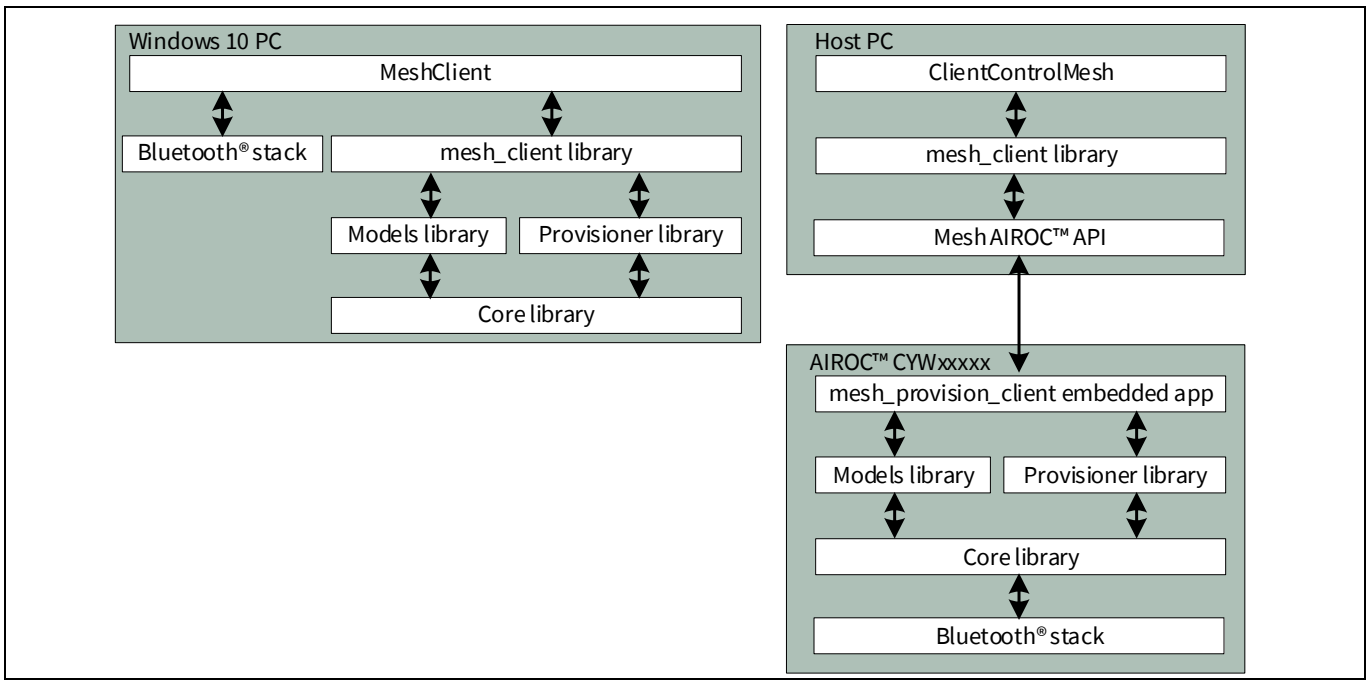



**Figure 2 MeshClient (left) and ClientControlMesh (right) software block diagram**

The MeshClient application uses the Bluetooth® stack as it exists on Windows 10. It uses a GATT Proxy connection [1] to control the Mesh. The ClientControlMesh application uses the Bluetooth® stack of the Infineon 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 model. 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 exercise Bluetooth® functionality such as starting Bluetooth® LE 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 the Bluetooth® stack of the OS. On the other hand, in the ClientControlMesh application, the MeshClient library uses AIROC™ Mesh Core, Models, and Provisioner library APIs to control embedded applications to perform all the Mesh-related work.

The AIROC™ Mesh Core, Models, and Provisioner 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”. MeshClient/ClientControlMesh applications perform as a Provisioner in the Mesh network. These applications maintain the database for the network, initiate a scan for unprovisioned devices, and perform the provisioning procedure as defined in Mesh Profile specification **[1]**. As a result 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 such as the network key and IV (Initialization Vector) 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 Advertising Bearers and 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 node of the Mesh network. The following is a partial list of things that the Provisioner must 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 determine 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 the GATT Proxy feature or Friend role feature, the Provisioner needs to specify if the node should use the feature.
- Add security keys. Network Keys (NetKey) if the node should also be a part of other subnets, and Application Keys (AppKey) for use with various Mesh models.
- Bind appropriate AppKeys to appropriate models of the new node. For example, the Provisioner can specify one AppKey to be used to configure a light bulb and a different AppKey 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 an 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 massages 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 applications 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 the 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.

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 an acknowledged message, each device in the group would send a reply. The Mesh stack monitors how many replies have been received; if a reply is not received from a specific node, 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 multiple bulbs as servers, and an ON/OFF switch as a client. 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 on 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, you should use the MeshClient application because it does not require an external device to run the Bluetooth® stack.

The user interface of the MeshClient and the ClientControlMesh applications are very similar. The only key difference between the two applications is the serial port selection and baud rate setting. These fields are not available in the MeshClient app because it uses the PC's built-in Bluetooth® stack. The ClientControlMesh application communicates with an external Infineon AIROC™ Evaluation board/device over the HCI UART. Therefore, these fields are provided in the ClientControlMesh application.

See the following section to learn how to select the serial port and baud rate.

*Note:* *From this point on, screenshots in this document are of the MeshClient app because most of other fields and buttons are similar in both apps.*

## 4.1 Creating and opening a Mesh network

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

**Step 1:** Program one evaluation board with the "mesh_provision_client" snip application.

This application can be created in ModusToolbox™ software using the Project Creator tool (**Quick Panel** > **New Application**). Select a BSP and then choose the "Mesh Provision Client" application. After project creation, it will be available in the IDE Project Explorer pane.

Once the board is programmed and connected to the PC, check the serial port number for the HCI UART. Do the following to check the COM port number:

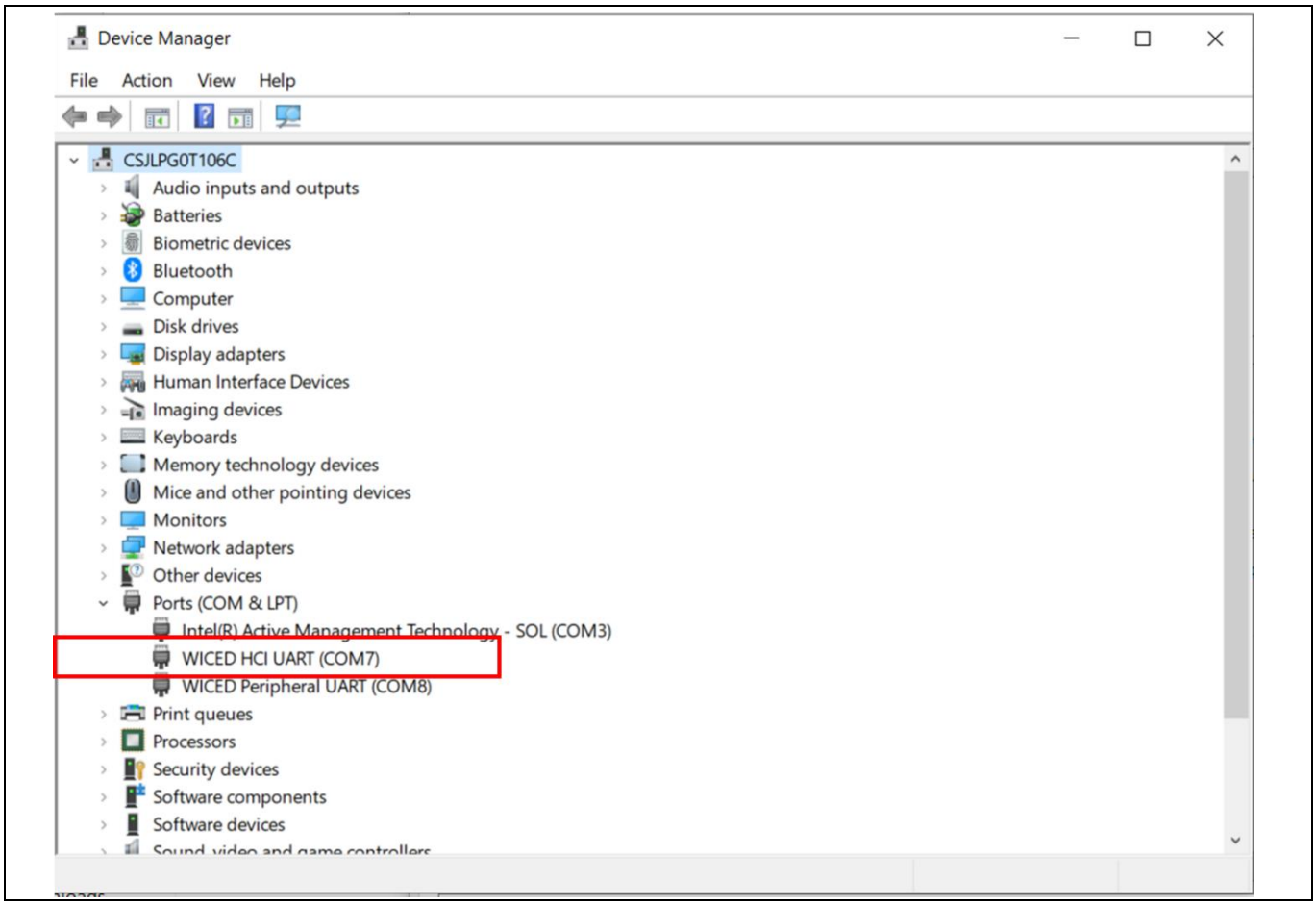
- Windows: In **Device Manager,** expand **Ports (COM & LPT)** and locate **WICED HCI UART**.
- Linux: Determine the serial port as **/dev/ttyWICED_HCI_UARTx**.
- macOS: Determine One of the **/dev/tty.usbserial-xxxx** devices (macOS does not provide distinct device names for HCI and Peripheral UARTs).

You can also find the HCI UART that was used when programming the application in the ModusToolbox™ software console output, for example:

```
Detecting serial port ...
Found serial port : COM10
```

This is the serial port to be used in the ClientControlMesh application. See the following screenshot.

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



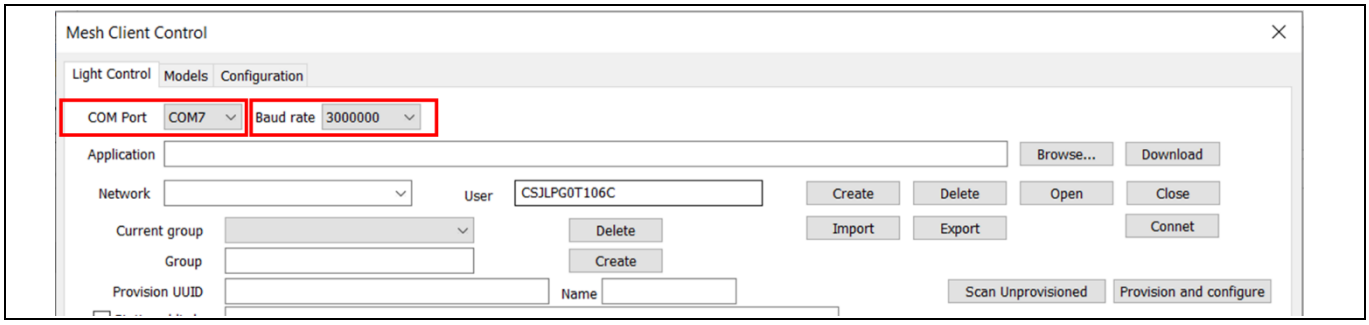

Once the COM port is identified, open the ClientControlMesh application.

*Note:* *If the board is connected to the PC after opening the ClientControlMesh application, the ClientControlMesh application will not detect the COM port. Therefore, make sure that you open the application **after** the board is connected and displayed in Device Manager on Windows, or seen under /dev in the filesystem on Linux or macOS.*

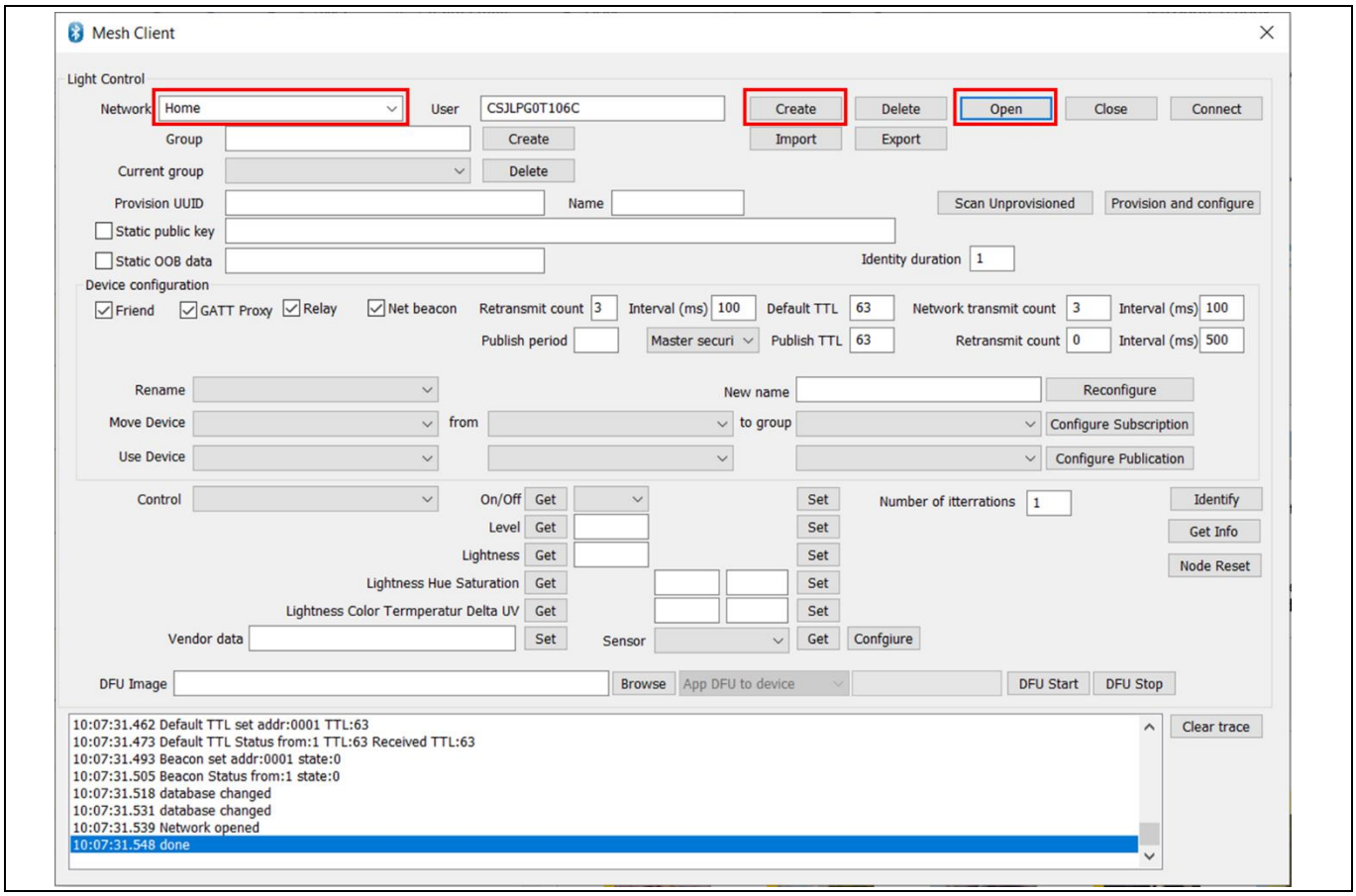
Select this HCI-UART COM port from the **COM port** drop-down menu.

Then, select **Baud rate** as 300,000 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 the “Mesh Provision Client” 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, and 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 the corresponding document from Bluetooth® SIG [3].

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

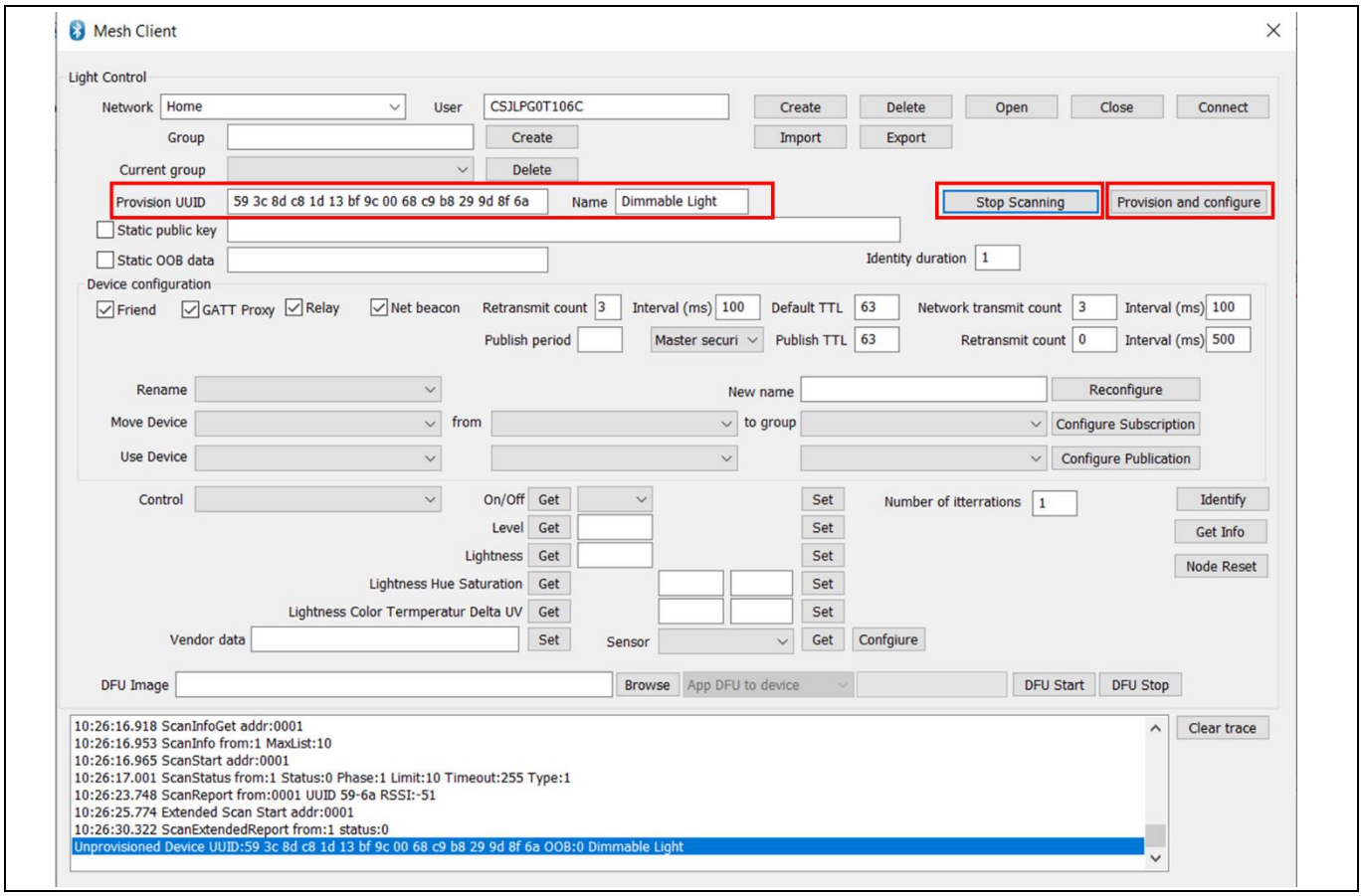
When you click **Open**, the MeshClient configures the stack with the parameters of the selected network. Similarly, the ClientControlMesh talks over the selected COM port to configure the stack running the "Mesh Provision Client" application on the embedded platform. The “done” trace at the end of the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

## 4.2 Adding a node

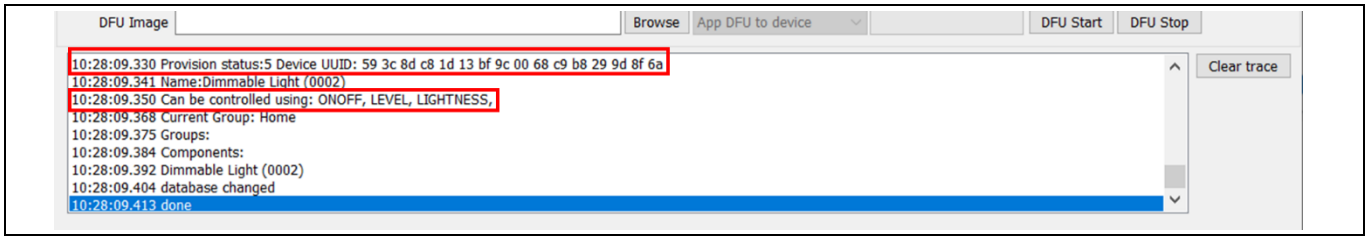
Use ModusToolbox™ software to build and download one of the mesh samples to an AIROC™ evaluation board. In the following description, the “Mesh Demo Dimmable Light” code example is used. See the respective kits’ user guide/getting started guide to learn how to program the board.

Do the following to provision a new node; see the following screenshot:

1. In the MeshClient window, click **Scan Unprovisioned**.
   The title of the button changes to **Stop Scanning** to indicate that the scan is active.
   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.
2. When you see the device that you want to work with, click **Stop Scanning**.
3. Click **Provision and configure**.



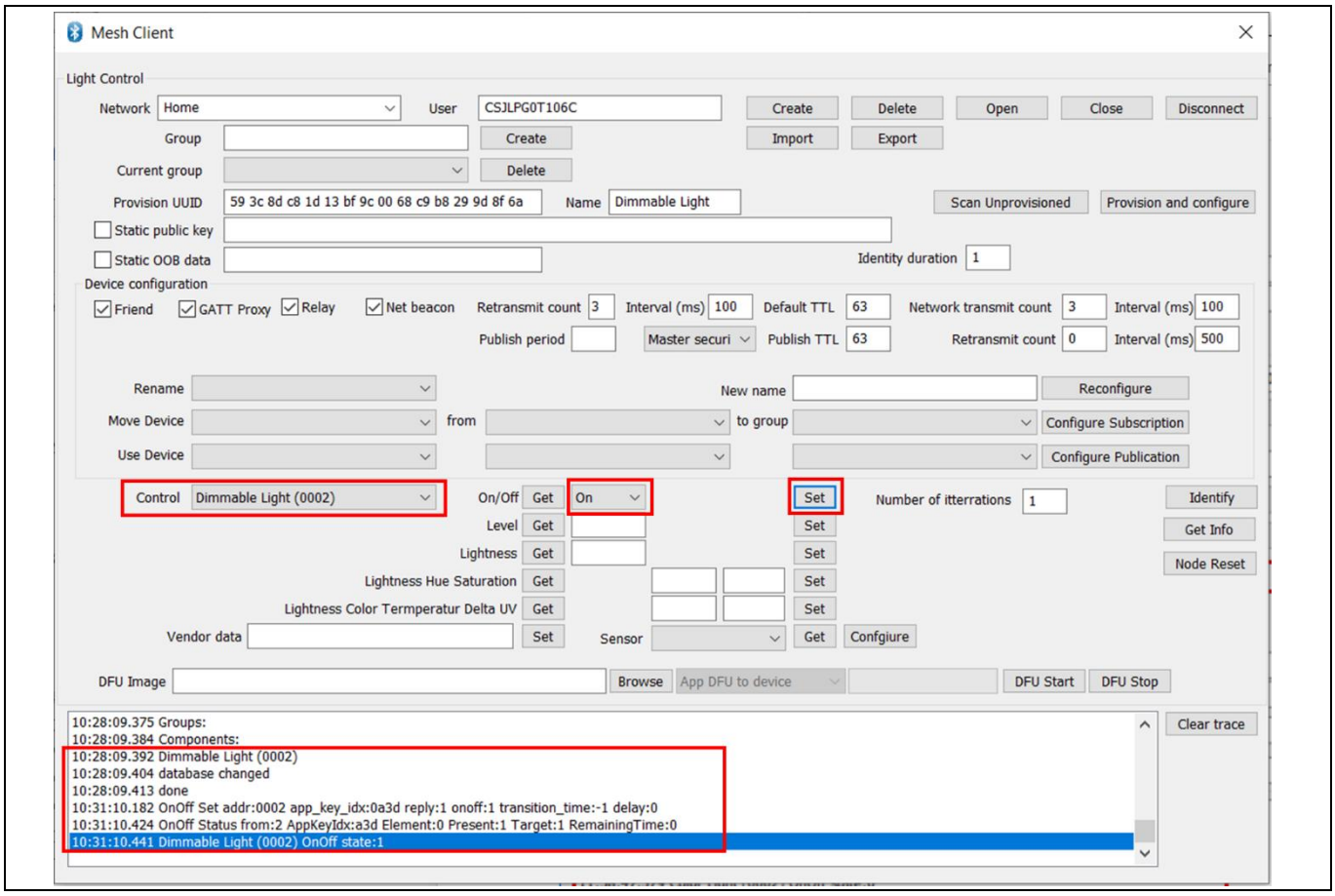

The provisioning and configuration process consists of several steps. While the process is happening, the status is displayed in the trace window. At the end of the process, **Provision status:5** appears in the trace window, indicating that the process has been completed successfully. The MeshClient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



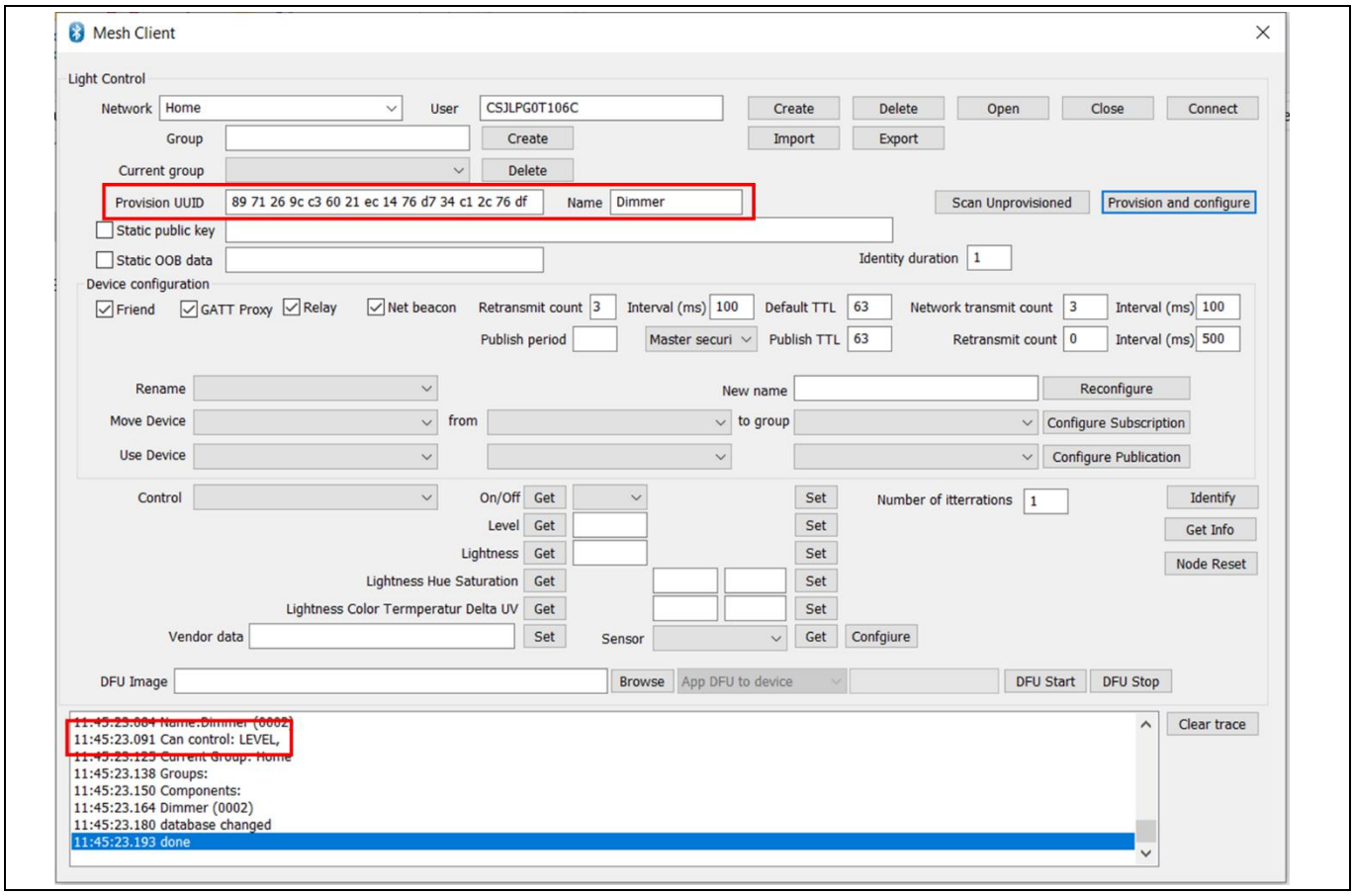

The trace window will print out the results of the operation.

If the device has been configured to be a GATT Proxy, the MeshClient will keep the connection to the new device open. If required, 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 in the **Control** dropdown and issue a **Get** command to retrieve the current **On/Off** status, or set a desired state to **On** by issuing a **Set** command. Before sending any command to the node, ensure that the app is connected to the proxy node i.e., the **Connect/Disconnect** button must show **Disconnect**.

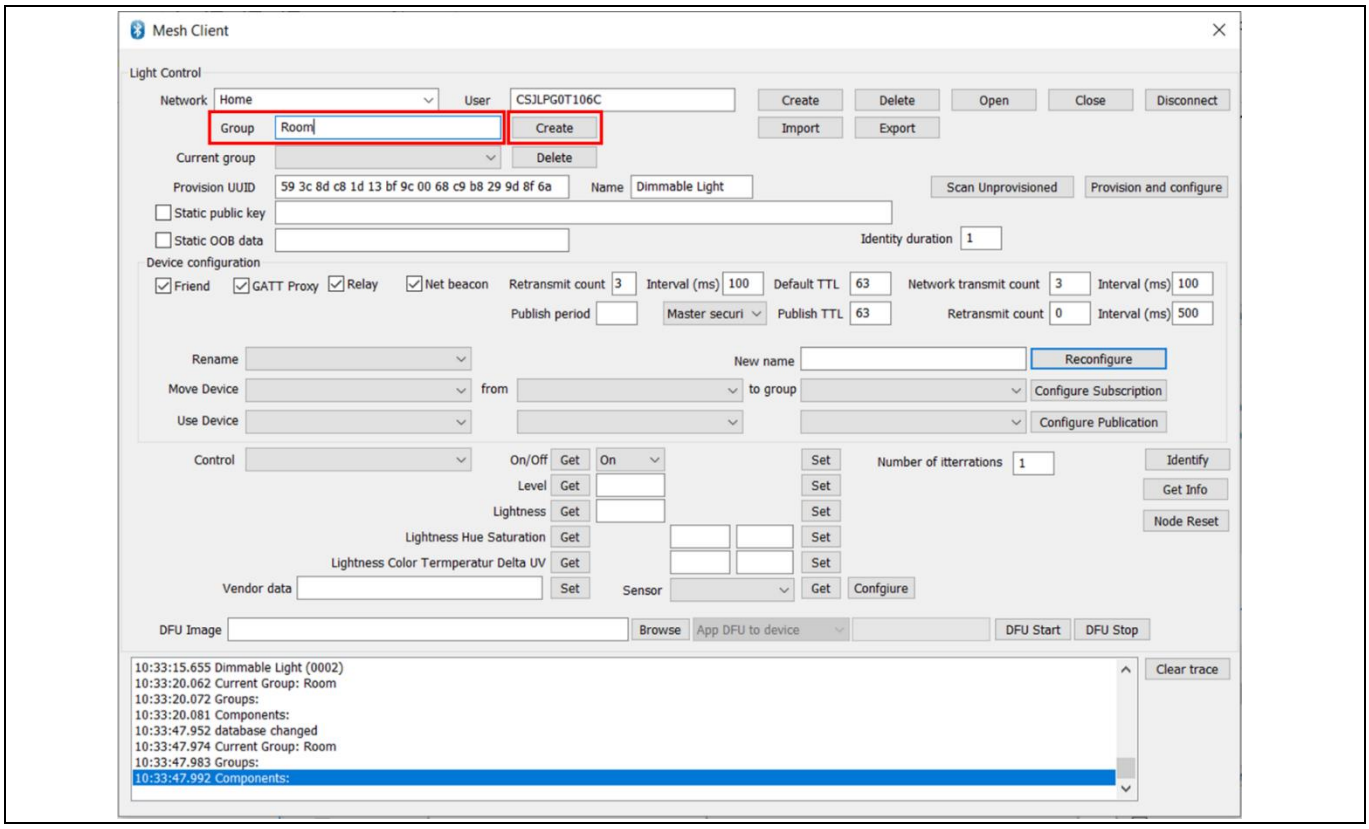



When a switch/dimmer or any other client is provisioned, instead of the “Can be controlled using” tag, the trace window will show “Can control”. For example, the following screenshot shows the “Can Control” trace when a dimmer (level client model) is provisioned. Because it is a level client, it can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


Mesh Client
Light Control
Network
Home
User
CSJLPG0T106C
Create
Delete
Open
Close
Connect
Group
Create
Import
Export
Current group
Delete
Provision UUID
89 71 26 9c c3 60 21 ec 14 76 d7 34 c1 2c 76 df
Name
Dimmer
Scan Unprovisioned
Provision and configure
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 securi
Publish TTL
63
Retransmit count
0
Interval (ms)
500
Rename
New name
Reconfigure
Move Device
from
to group
Configure Subscription
Use Device
Configure Publication
Control
On/Off
Get
Set
Number of itterrations
1
Identify
Level
Get
Set
Get Info
Lightness
Get
Set
Node Reset
Lightness Hue Saturation
Get
Set
Lightness Color Termperatur Delta UV
Get
Set
Vendor data
Set
Sensor
Get
Confgiure
DFU Image
Browse
App DFU to device
DFU Start
DFU Stop
11:45:23.091 Can control: LEVEL,
11:45:23.138 Groups:
11:45:23.150 Components:
11:45:23.164 Dimmer (0002)
11:45:23.180 database changed
11:45:23.193 done
Clear trace

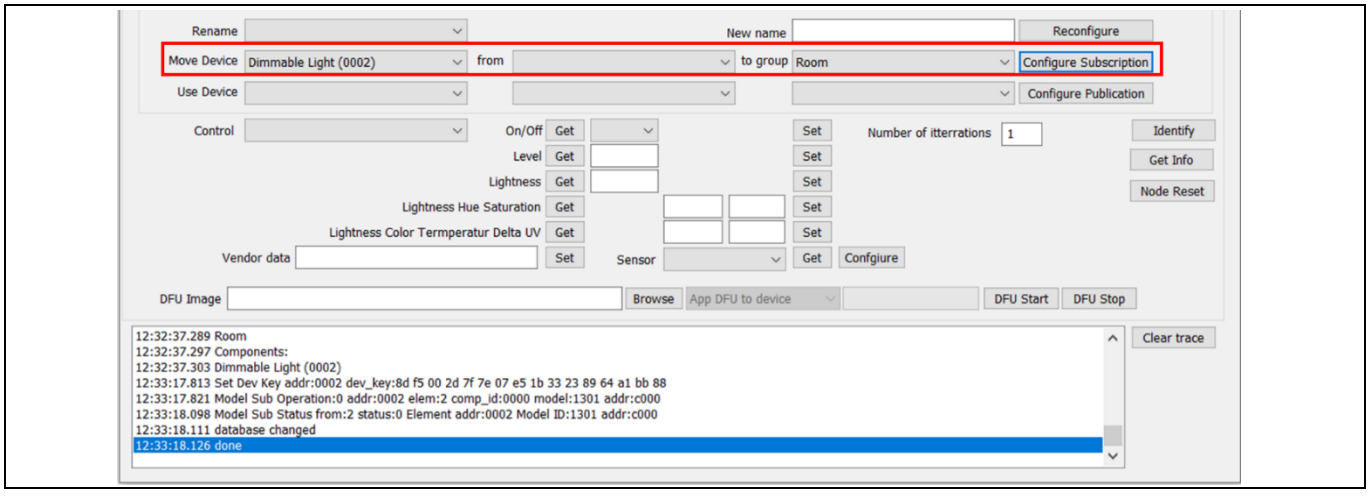
## 4.3 Creating and managing groups

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.



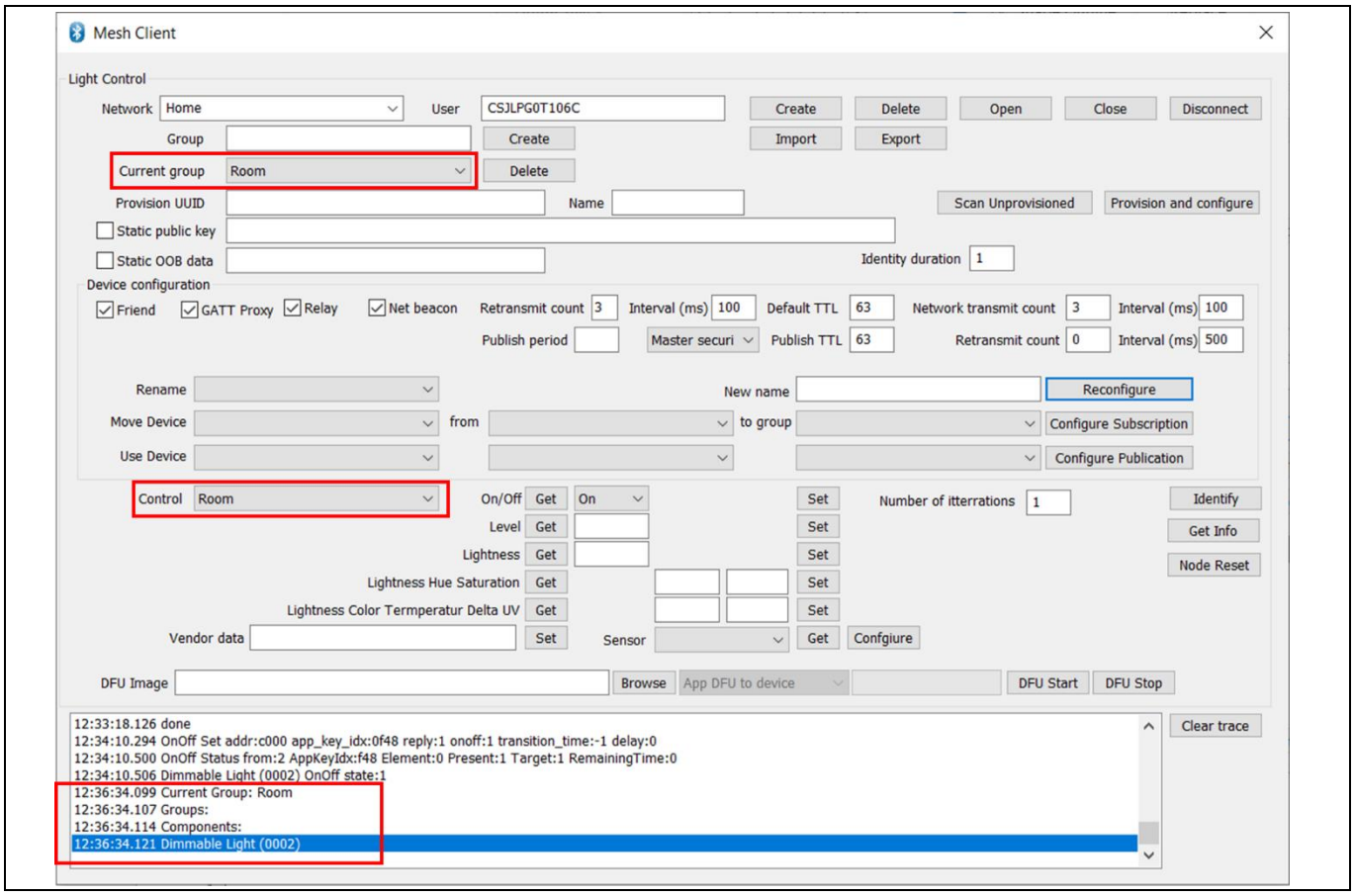

You can then select the group in the **Current group** field. The nodes now will be provisioned into the created group.

You can also move devices between previously created groups. The **Configure Subscriptions** button allows you to put the device into a created group, where the device is now subscribed to process unicast messages destined to that device as well as messages addressed to the group.

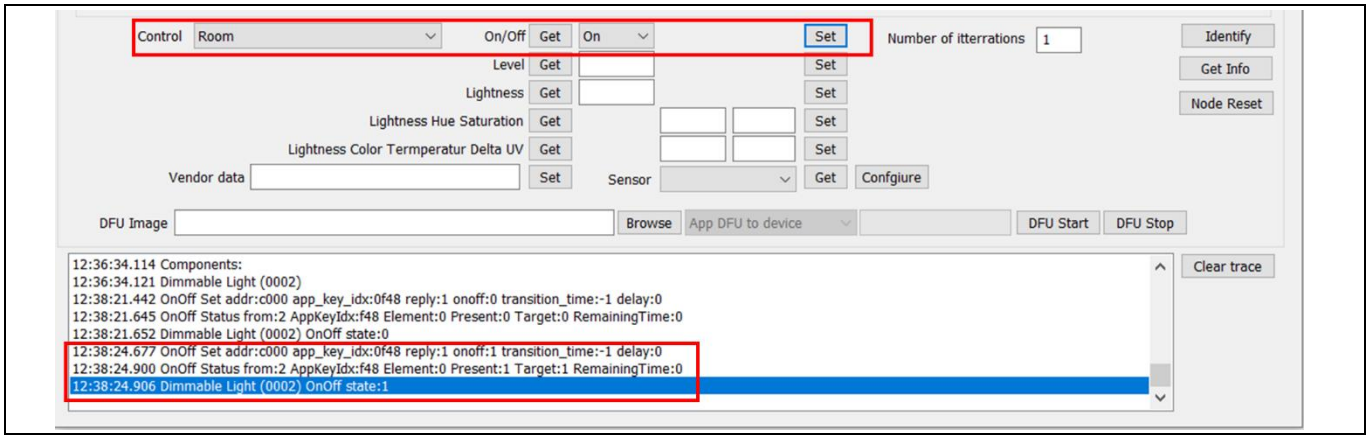


If a new device is provisioned while the current selected group is 'Room', the device will be assigned to this group, without needing to perform the 'Move' operation. In the **Control** field, you can select an individual device, a group address, or the name of the network to unicast, multicast, or broadcast Mesh control messages respectively.

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

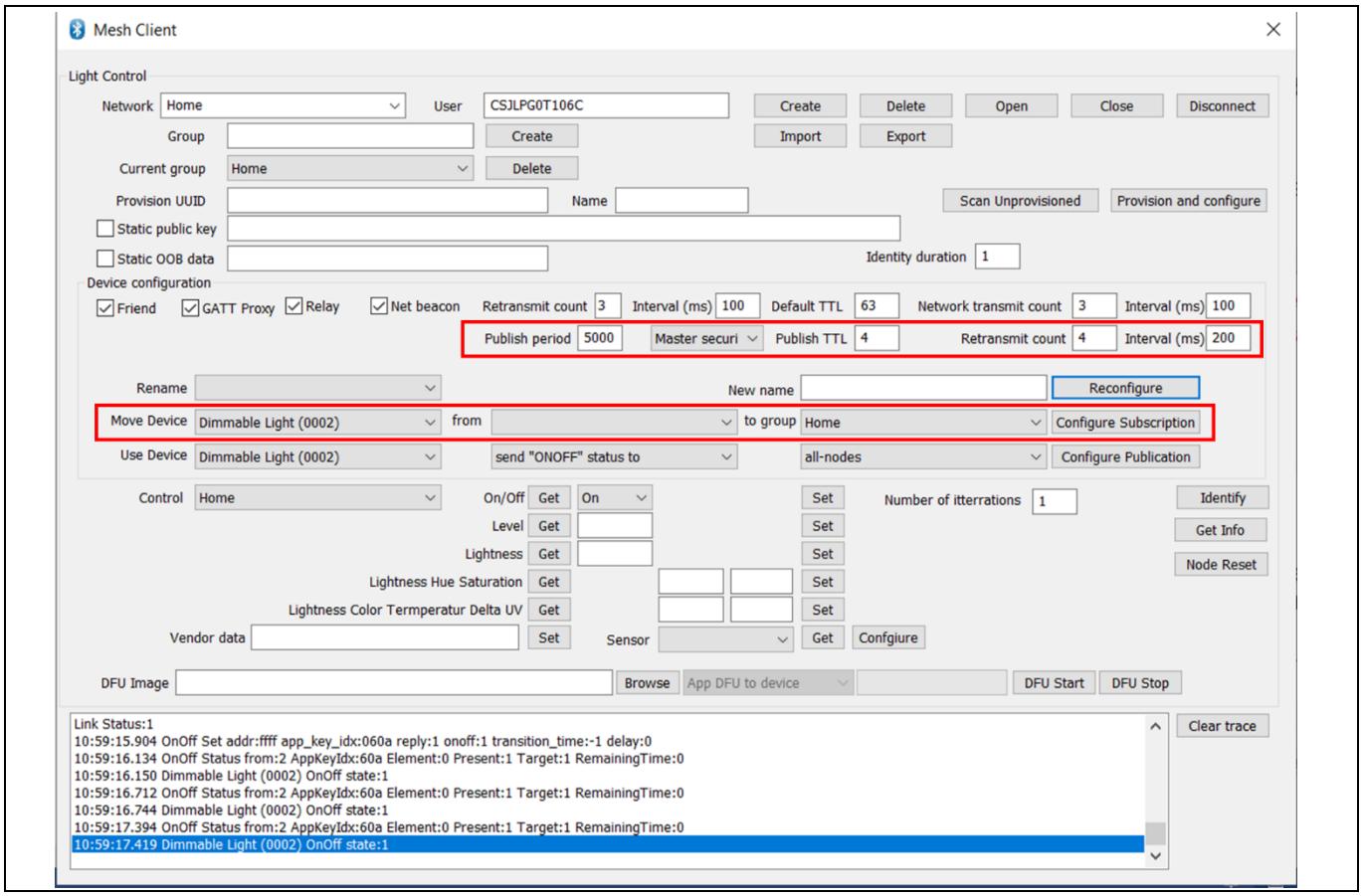



The entire group can be controlled by using a single command. Select the group name to be controlled in **Contro**l field, select the action, and click **Set**.



## 4.4 Configuring devices

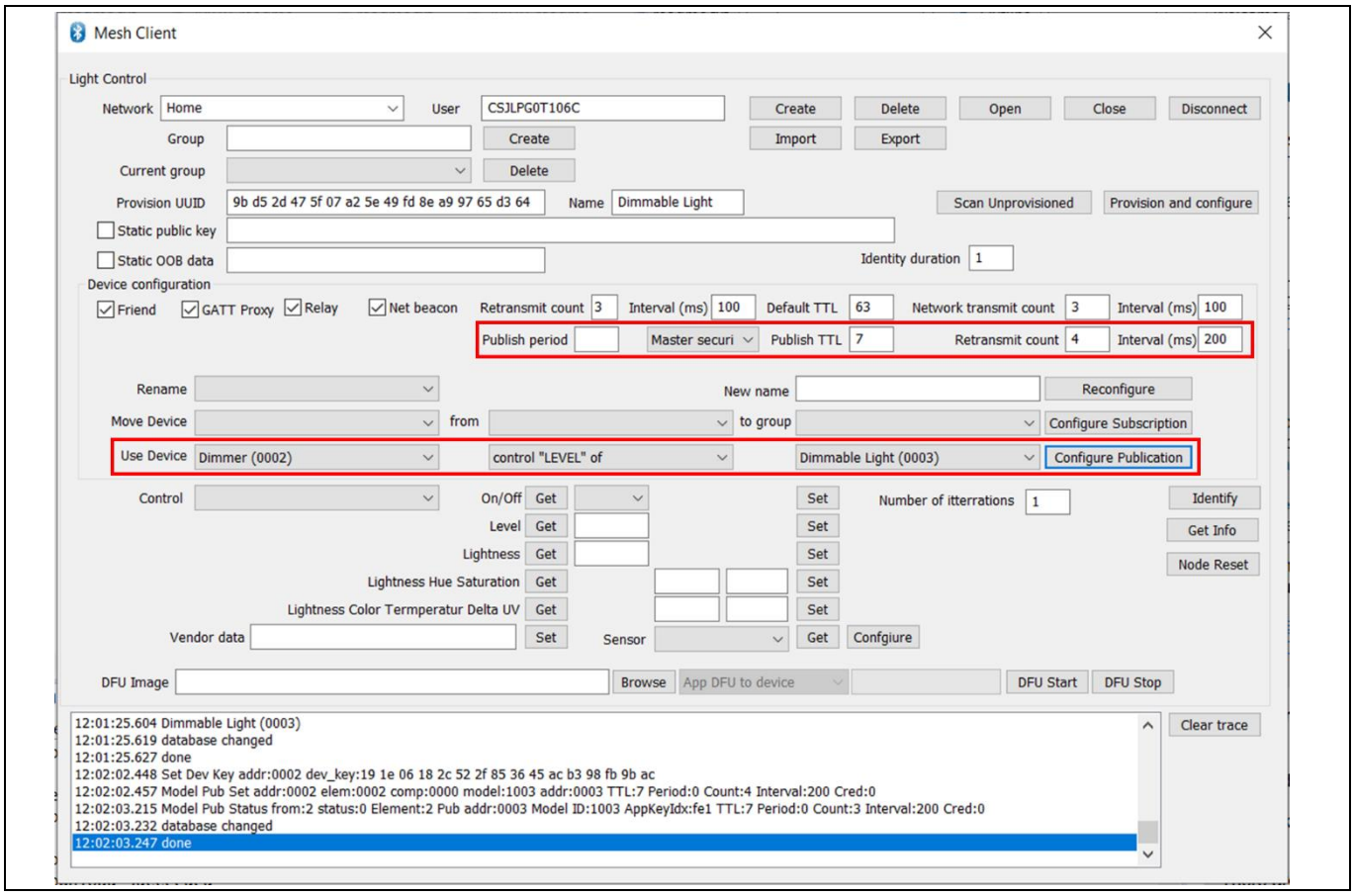
The controls in the “Device configuration” section allow you to configure multiple parameters for sending or relaying messages. By default, parameters from the initial provisioning are used; you can specify a new name for an already configured device, select the device in the **Rename** field, and click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




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

Similarly, a device can be configured as a controlling device 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 method 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 the 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.

## 4.5 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 click **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
```

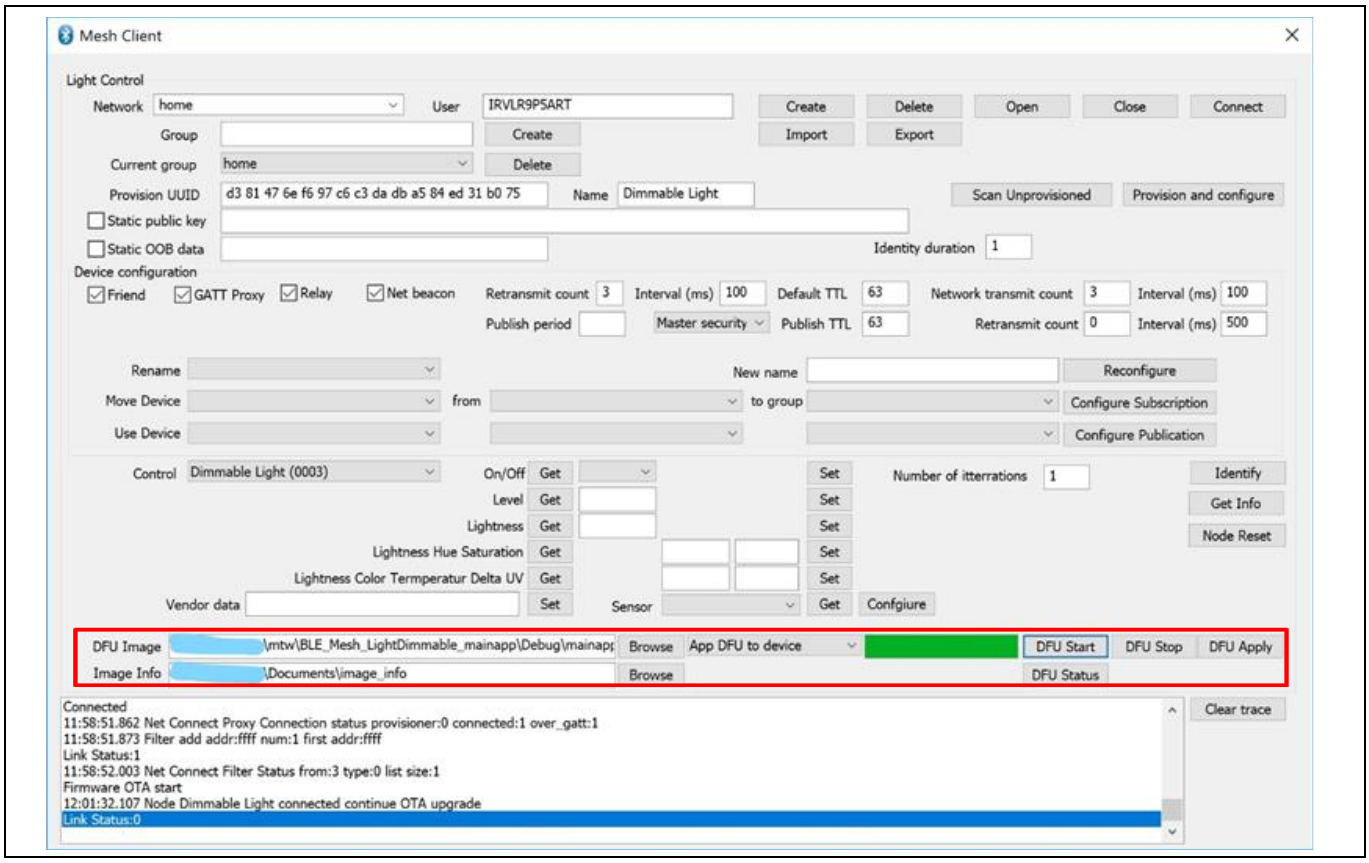



**Figure 3 Mesh client DFU**

# 5 Mesh performance testing

## 5.1 Overview

Common performance indicators that must be tested to ensure 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 a turn ON/OFF lights command is issued to all bulbs in the network, how many of the nodes turn ON/OFF?
   b. When vendor-specific data is sent to a destination node, in how many of the N 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. The key performance indicators for the Mesh performance are reliability, latency, power consumption, and network throughput. The variables that 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 | Directly |
| **Power consumption** | Directly | Directly | Directly | Directly | Directly |
| **Network throughput** | Inversely | Directly | Inversely | Directly | Inversely |

### 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 node is 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.

The 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. The 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 set up 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)

The 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 in the least number of hops and thus decreasing the power consumption, latency and improving the network throughput. The TTL value is configured on the **Light Control** tab of the application.

For networks with a 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 network performance.

### 5.2.4 Power consumption

Power consumption at each node is affected by several operating parameters of the node. One of the parameters 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. The 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 AIROC™ 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 the Mesh Performance Testing 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. A few common reasons that may 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 suggest increasing the Tx Power

    Too many nodes may suggest lowering the Tx Power
- Interference from surroundings like walls, structures, and so on
  - Obscured or heavily hindered locations may require a node to have higher Tx power to function effectively
  - A lower Tx power setting may be necessary in situations where the range of the node needs to be restricted to a small area.

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

The 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, the 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.

The 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 are 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 set up to improve the Mesh performance indicators, namely the latency, number of hops, power consumption, and network throughput.

The 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 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

The evaluation of network-level Mesh performance indicators will require the evaluation of the 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, and the 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 – AIROC™ 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. The MCC application will collect and log Mesh application state events from the Mesh node to which it is connected. Multiple instances of the MCC application 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 the MCC application may be set up 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 set up 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 AIROC™ board running the Mesh Application Client (for example: ON/OFF client, vendor client, and so on).
3. Attach the MCC with UDP clients to the AIROC™ 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 time stamp. At this point, logs from one Mesh Application Client and the Mesh Application Servers are consolidated, as shown in **Figure 4**.

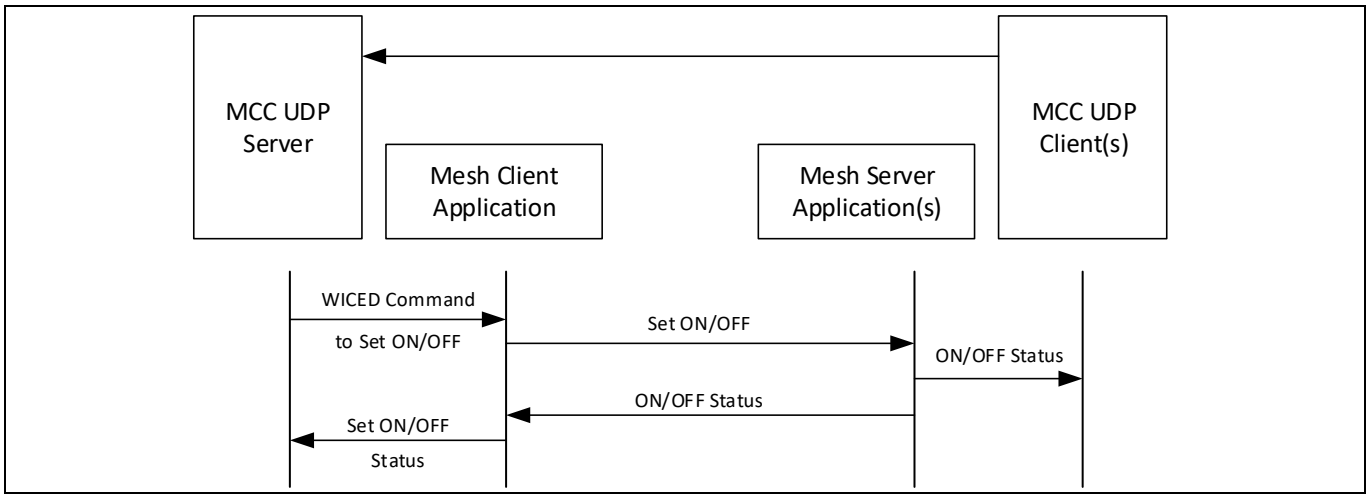



**Figure 4 Direct/one-way approach**

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 millisecond accuracy
- Ability to account for latency of logging information using UDP protocol over the local Wi-Fi 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. The 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 the number of hops based on the change in the TTL values, a customized Mesh VendorClient/Server can be used. VendorClient will send a message to a custom VendorServer implementation in the mesh_perf_testing application with a certain TTL value as set in the node configuration.

- Packets traverse across the Mesh network and reach the VendorServer.
- When the VendorServer 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**.

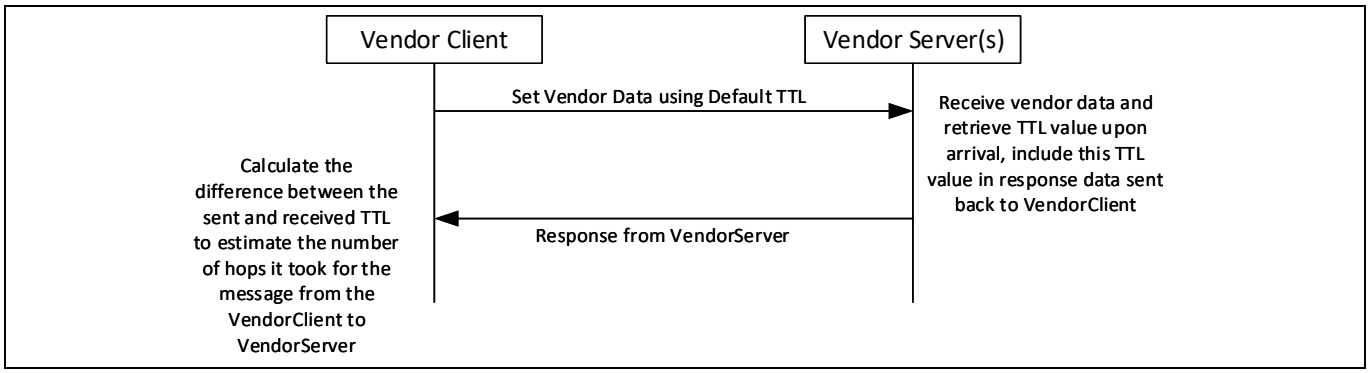



**Figure 5 Indirect/round-trip approach**

## 5.3 Testing procedure

Use latest the ModusToolbox™ 2.2+ software and any AIROC™ platform that supports Mesh, such as CYW20819, CYW20735, and so on, and install the latest BTSDK. At the time of publication of this document, BTSDK 2.9.0 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_btsdk\tools\btsdk-host-apps-mesh\<branch>\VS_ClientControl\Release\ClientControlMesh.exe*

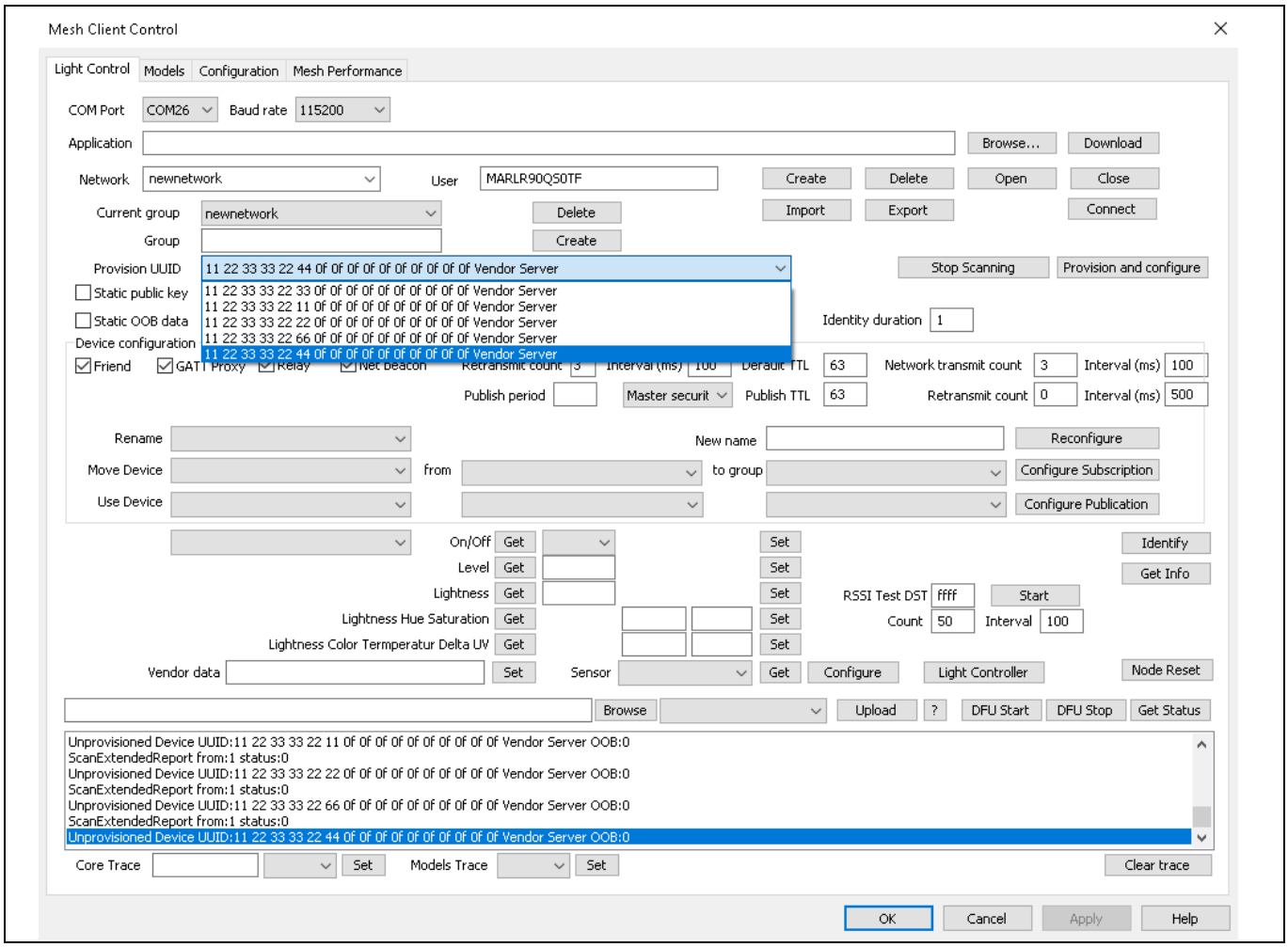
Launch *ClientControlMesh.exe* from the command line as shown below, and open the WICED HCI UART (usually the first COM port) and connect to the VendorClient. To enable tracing to the file, add `-T` to the command line:

```
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 Tera Term and connect the WICED Peripheral UART port (usually COM port with a greater value, 921,600 baud) to PUART on VendorClient/VendorServer and make sure traces are received.

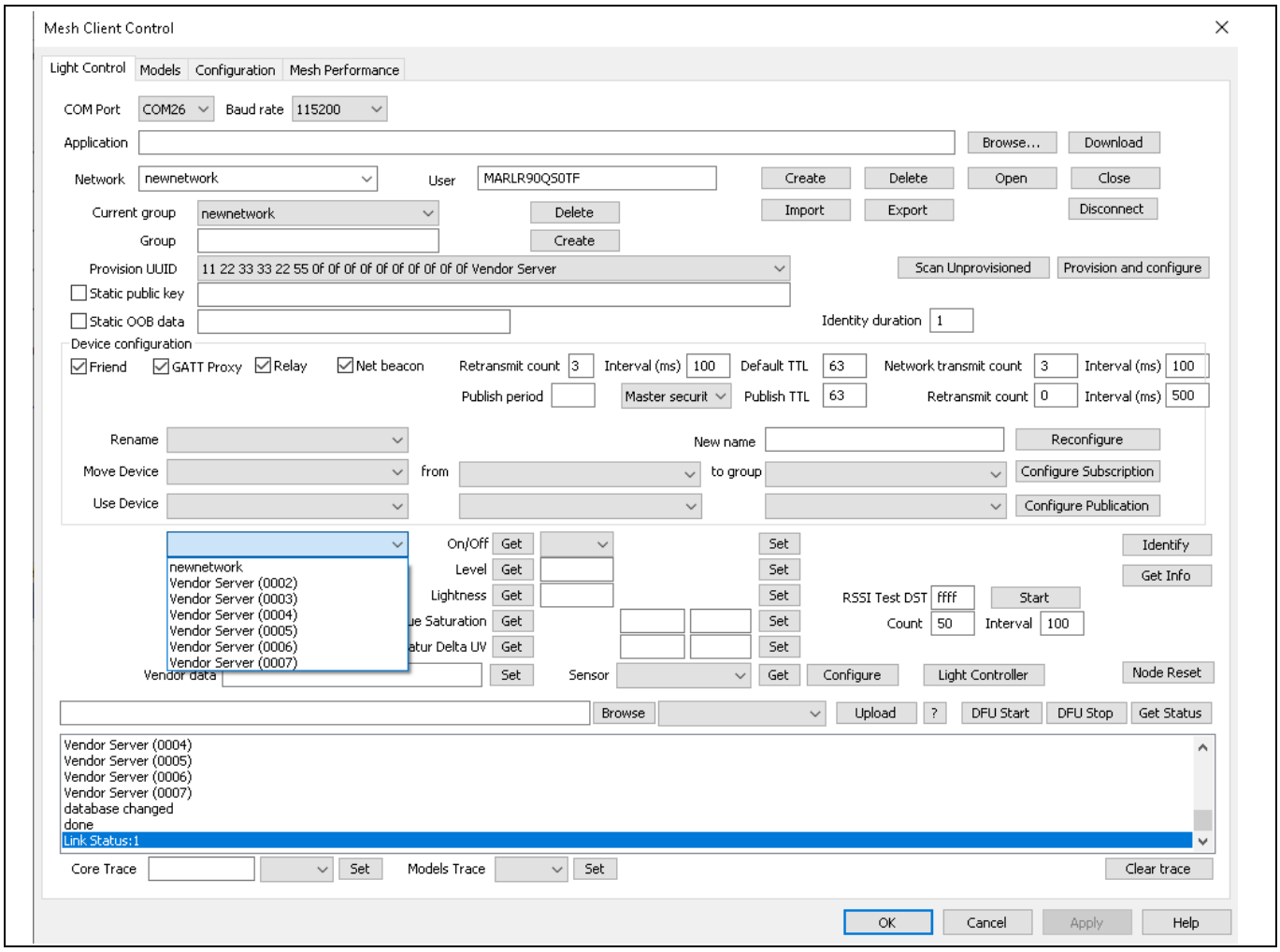
Follow these steps on the ClientControlMesh – Light Control tab to test the reliability of sending and receiving data between VendorClient and VendorServer:

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.




5. Click **Provision and Configure** to provision the device, and wait for the provisioning to 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 example, (0002) and so on.




Upon provisioning all devices, place the devices in the mesh network topology to be used for the testing. See **Figure 6**.

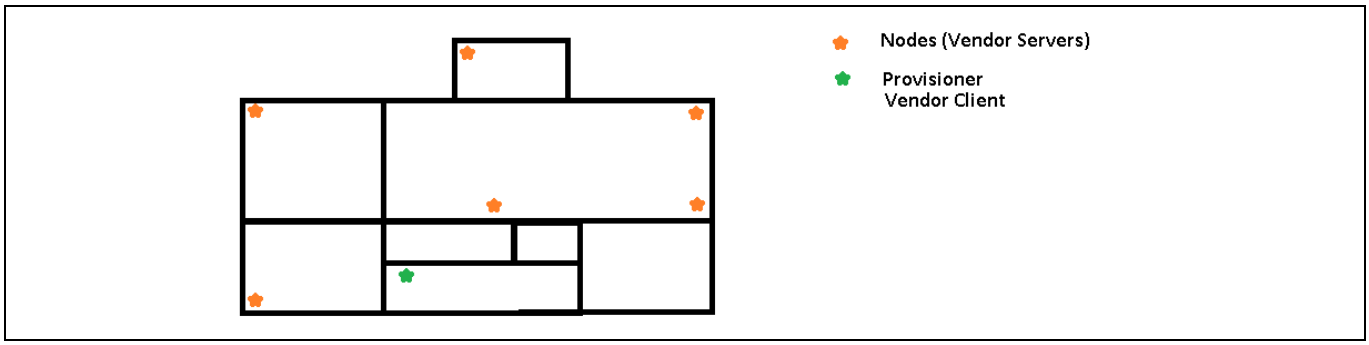
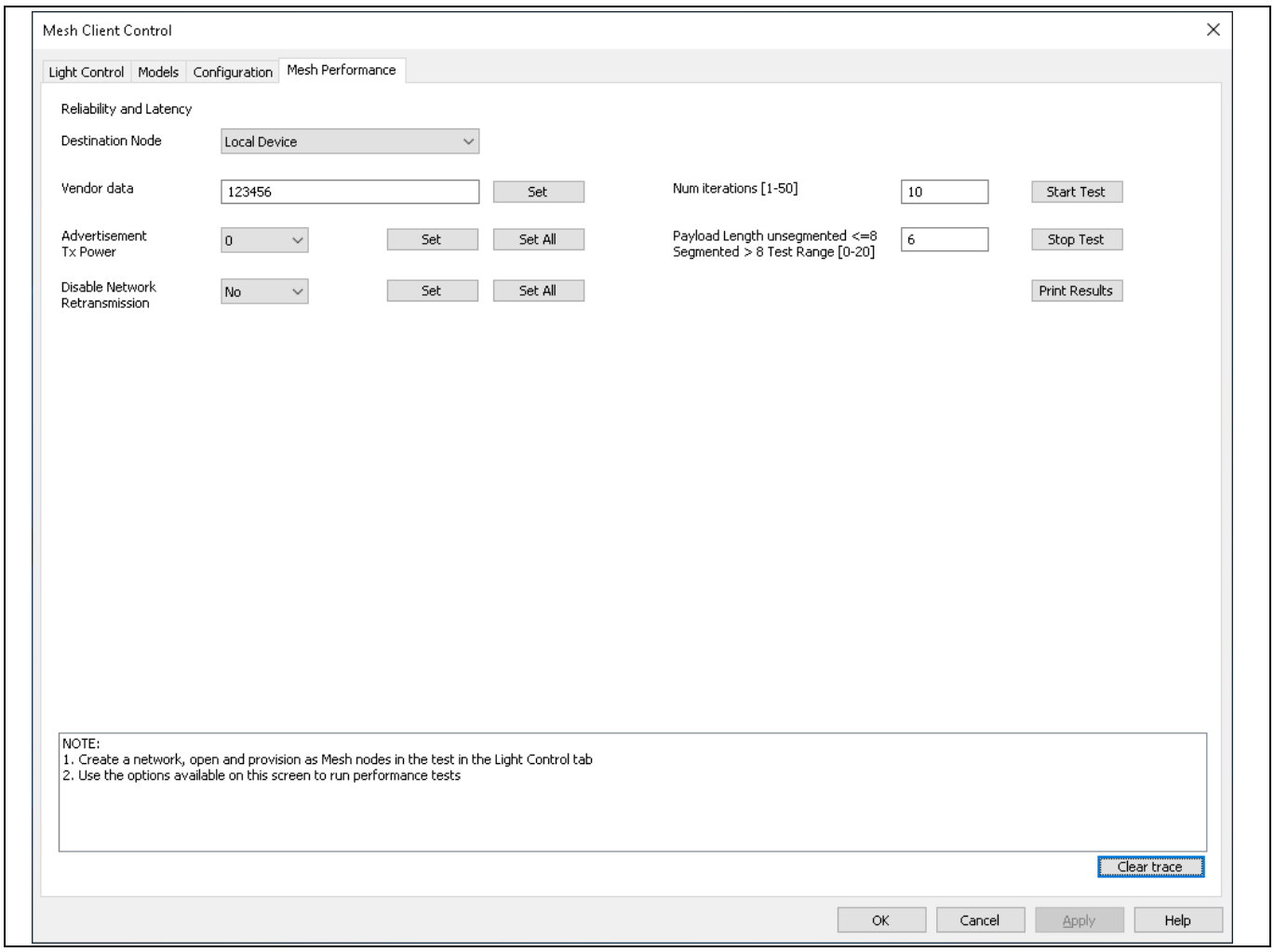



**Figure 6 Devices in Mesh network topology**

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.

The 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, the Mesh Performance tool does not support evaluation of the network throughput and power consumption.




On the **Mesh Performance** tab, you can perform tests to gather and understand the data on the reliability, latency, number of hops, and 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 the least power and 4 indicates the maximum power. Set the value to 0 if the Mesh network is dense (many devices in a small space) and make sure that you get at least a few hops.

### Testing the message reliability (one time)

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

### Testing the 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.

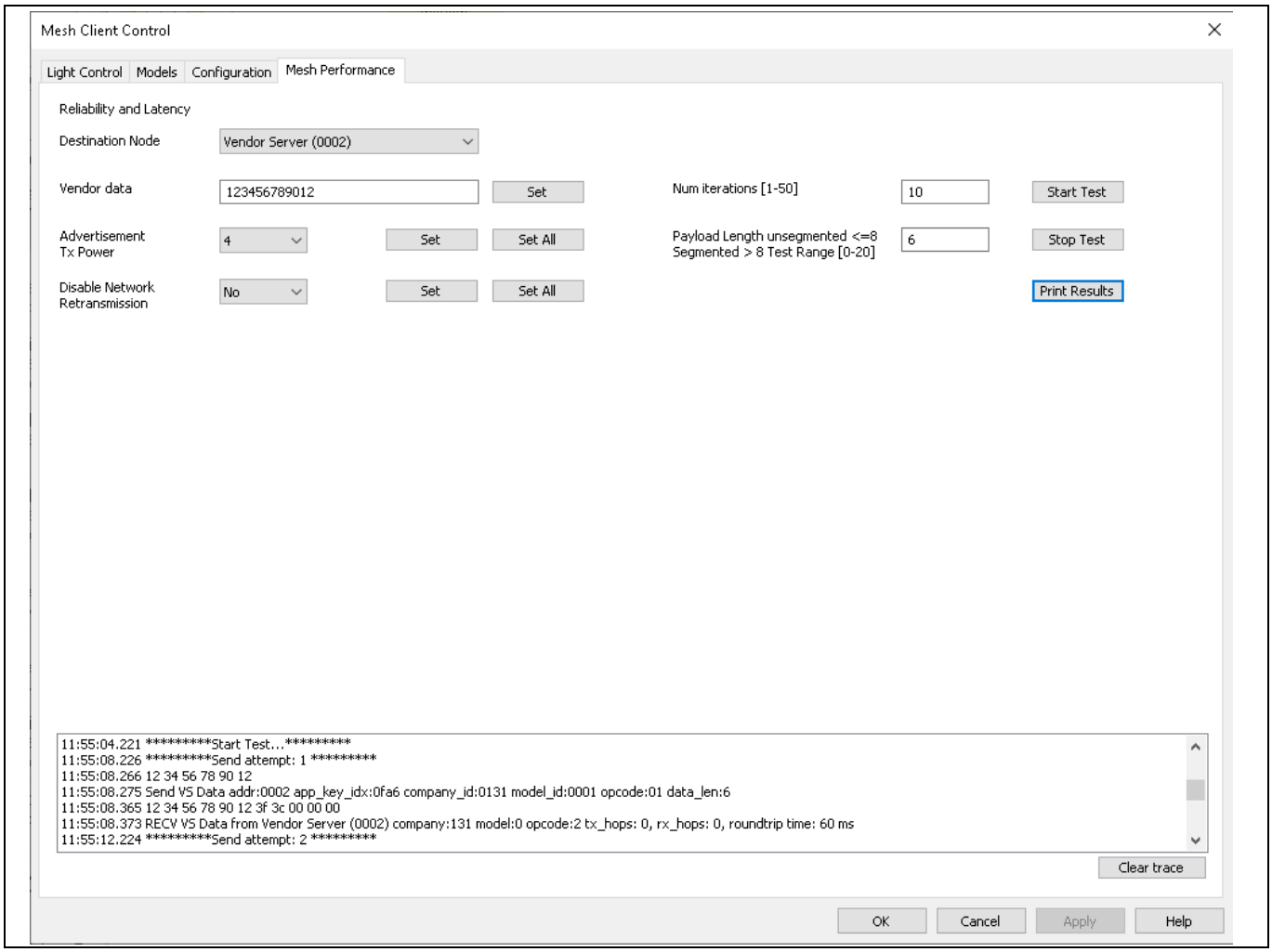
    The default test runs for 10 times with a payload size of 3 bytes.

### Testing the reliability with relay nodes in between (multiple nodes)

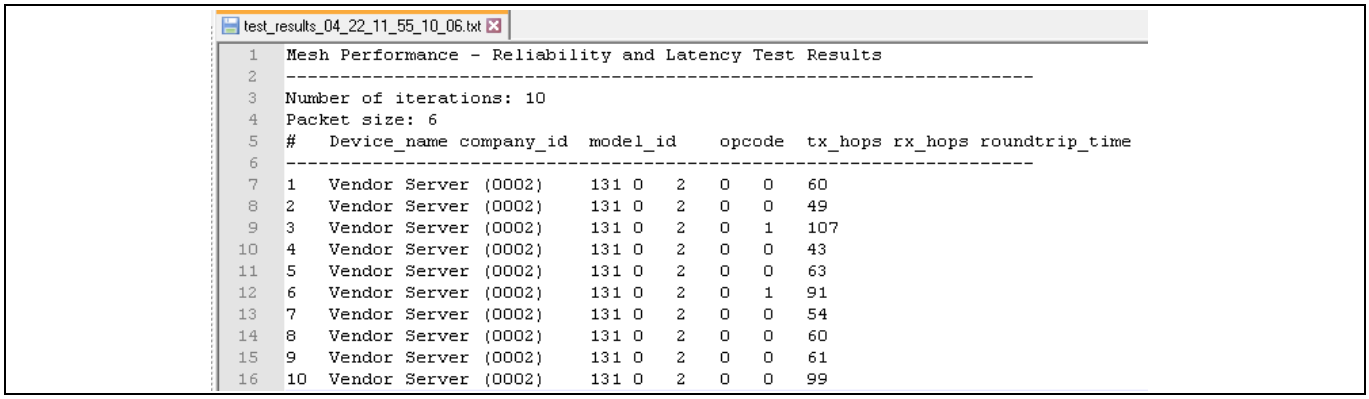
11. Select a destination device that is farther from the Provisioner node such that there are other VendorServers between the VendorClient and the destination VendorServer and the data sent must be relayed or must hop over multiple VendorServers to reach the destination device.

    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 indicate 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.

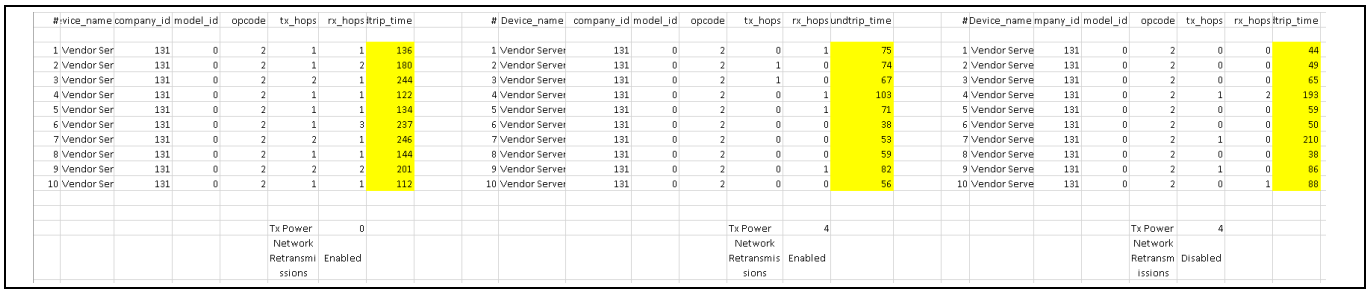



12. Click **Print Results** to print the results of multiple loop tests in to a tab-delimited file, which can be used for data analysis.



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
```



| #vice_name | company_id | model_id | opcode | tx_hops | rx_hops | trip_time |
|---|---|---|---|---|---|---|
| 1 Vendor Ser | 131 | 0 | 2 | 1 | 1 | 136 |
| 2 Vendor Ser | 131 | 0 | 2 | 1 | 2 | 180 |
| 3 Vendor Ser | 131 | 0 | 2 | 2 | 1 | 244 |
| 4 Vendor Ser | 131 | 0 | 2 | 1 | 1 | 122 |
| 5 Vendor Ser | 131 | 0 | 2 | 1 | 1 | 134 |
| 6 Vendor Ser | 131 | 0 | 2 | 1 | 3 | 237 |
| 7 Vendor Ser | 131 | 0 | 2 | 2 | 1 | 246 |
| 8 Vendor Ser | 131 | 0 | 2 | 1 | 1 | 144 |
| 9 Vendor Ser | 131 | 0 | 2 | 2 | 2 | 201 |
| 10 Vendor Ser | 131 | 0 | 2 | 1 | 1 | 112 |
| Tx Power | 0 | | | | | |
| Network Retransmi | Enabled | | | | | |

| # Device_name | company_id | model_id | opcode | tx_hops | rx_hops | undtrip_time |
|---|---|---|---|---|---|---|
| 1 Vendor Server | 131 | 0 | 2 | 0 | 1 | 75 |
| 2 Vendor Server | 131 | 0 | 2 | 1 | 0 | 74 |
| 3 Vendor Server | 131 | 0 | 2 | 1 | 0 | 67 |
| 4 Vendor Server | 131 | 0 | 2 | 0 | 1 | 103 |
| 5 Vendor Server | 131 | 0 | 2 | 0 | 1 | 71 |
| 6 Vendor Server | 131 | 0 | 2 | 0 | 0 | 38 |
| 7 Vendor Server | 131 | 0 | 2 | 0 | 0 | 53 |
| 8 Vendor Server | 131 | 0 | 2 | 0 | 0 | 59 |
| 9 Vendor Server | 131 | 0 | 2 | 0 | 1 | 82 |
| 10 Vendor Server | 131 | 0 | 2 | 0 | 0 | 56 |
| Tx Power | 4 | | | | | |
| Network Retransmis | Enabled | | | | | |

| #Device_name | mpany_id | model_id | opcode | tx_hops | rx_hops | trip_time |
|---|---|---|---|---|---|---|
| 1 Vendor Serve | 131 | 0 | 2 | 0 | 0 | 44 |
| 2 Vendor Serve | 131 | 0 | 2 | 0 | 0 | 49 |
| 3 Vendor Serve | 131 | 0 | 2 | 0 | 0 | 65 |
| 4 Vendor Serve | 131 | 0 | 2 | 1 | 2 | 193 |
| 5 Vendor Serve | 131 | 0 | 2 | 0 | 0 | 59 |
| 6 Vendor Serve | 131 | 0 | 2 | 0 | 0 | 50 |
| 7 Vendor Serve | 131 | 0 | 2 | 1 | 0 | 210 |
| 8 Vendor Serve | 131 | 0 | 2 | 0 | 0 | 38 |
| 9 Vendor Serve | 131 | 0 | 2 | 1 | 0 | 86 |
| 10 Vendor Serve | 131 | 0 | 2 | 0 | 1 | 88 |
| Tx Power | 4 | | | | | |
| Network Retransm | Disabled | | | | | |

**Figure 7 Sample results under different conditions**

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)

### 5.3.1 Application settings and configuration

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.*

**Code Listing 1 mesh_provision_client.c**

```
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
#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
```

The 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

[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 |
|---|---|---|
| ** | 2018-05-01 | Initial release |
| *A | 2019-04-24 | Removed Associated Part Family<br>Updated for BTSDK release |
| *B | 2019-10-15 | Updated for ModusToolbox™ software 2.0 |
| *C | 2020-02-11 | Updated for latest WICED Studio and ModusToolbox™ software changes |
| *D | 2020-06-10 | Added Mesh Performance Testing information |
| *E | 2021-02-26 | Updated for BTSDK 2.9.0 changes |
| *F | 2021-12-02 | Fixed typos, outdated information, and branding updates |
| *G | 2022-03-02 | Updated hyperlinks across the document. |

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

**Edition 2022-03-02**

**Published by**

**Infineon Technologies AG**

**81726 Munich, Germany**



**Do you have a question about this document?**

**Go to www.infineon.com/support**

**Document reference**

**002-26575 Rev. *G**

**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.