AN1352: Running the Bluetooth Mesh Interoperability (IOP) Test Application Note

Interoperability (IOP) is one of the key value propositions of Bluetooth Mesh and something that consumers have come to expect from Bluetooth enabled end products.

This document describes Silicon Labs IOP test framework comprising of hardware kits, embedded software, and mobile app. It also explains the requirements for building the IOP test setup, running the test, and collecting data for further analysis.

KEY FEATURES

- Software and Hardware requirements to run the IOP test
- Bringing up the test environment and running the test
- Collecting test data from EFR32 and mobile phone

Table of Contents

1. Introduction

IOP is a cornerstone of Bluetooth and one of the key reasons why this wireless technology has become ubiquitous. It enables end users to mix and match devices between different vendors without fearing connectivity issues. For example, whether a light switch from a company A will connect to a light bulb from a company B to control the lightning.

It is therefore essential that when a customer is looking for a Bluetooth solution supplier for his design, the supplier can provide means to test IOP between his Bluetooth solution and 3rd party devices.

One of the most common use cases for Bluetooth enabled devices is interaction with smartphones where a mobile app is used for command and control of the Bluetooth device. This use case places IOP in the spotlight because of the large number of permutations between smartphone hardware (namely Bluetooth chipset), low level firmware (typically BLE link layer), mobile OS (typically BLE host stack), and mobile OS version.

Silicon Labs provides a framework to test IOP between the EFR32 family of SoCs and a large number of smartphones currently on the market. This framework is used to run IOP testing against a large list of devices periodically. AN1308 contains both IOP test results and the IOP test plan.

Subsequent sections list the requirements for the IOP test framework. Subsequent chapters describe how to bring up the test environment, run the IOP test, and collect data for further analysis.

1.1 Hardware Requirements

The IOP embedded software is available for the following radio boards:

- BRD4104A (based on the xG13 SoC)
- BRD4181A (based on xG21 SoC) – currently in obsolete state and may no longer be available for purchase. However, if you have already purchased this board, it can be used for IOP testing
- BRD4181B (based on xG21 SoC – a new version of BRD4181A with a corrected pinout, more details in the PCN.
- BRD4182A (based on the xG22 SoC)

You can purchase the following boards stand-alone or as a part of a starter kit.

- BRD4104A can be purchased stand-alone with OPN SLWRB4104A or as part of SLWSTK6020B starter kit
- BRD4181B can be purchased stand-alone with OPN SLWRB4181B or as part of SLWSTK6006A starter kit
- BRD4182A can be purchased stand-alone with OPN SLWRB4182A or as part of SLWSTK6021A starter kit

1.2 Software Requirements

The IOP embedded software is available in the Silicon Labs GSDK starting with version 3.2.0 and later. Users should install Simplicity Studio 5 and Bluetooth SDK 3.2.0 or newer, which is part of GSDK 3.2.0. For more information about installing Simplicity Studio 5, see QSG169: Bluetooth® SDK v3.x Quick-Start Guide.

1.3 Interoperability (IOP) Test Demos

The interoperability tests check whether the Bluetooth Mesh Stack, which is running on the board is compatible with a smartphone or not. These are test procedures containing several test cases for each Bluetooth Mesh operation. These demos are meant to be used with the Bluetooth Mesh app and are currently supported only by SLWRB4104A EFR32BG13, SLWRB4181A EFR32MG21, SLWRB4181B EFR32MG21, and SLWRB4182A EFR32MG22.

- Bluetooth Mesh - IOP Test - Friend node: Friend example for the IOP test. This node acts as a friend for the low power node and caches messages sent to it when the low power node is sleeping.
- Bluetooth Mesh - IOP Test - LPN node: Low-power node example for the IOP test. This node acts as a typical low power device and sleeps most of the time. It needs a friend node to cache messages and forward them when polled.
- Bluetooth Mesh - IOP Test - Proxy node: Proxy example for the IOP test. This node forwards/relays messages between GATT and advertising bearers in the network.
- Bluetooth Mesh - IOP Test - Relay node: Relay example for the IOP test. This node acts as a relay, i.e., if a node is out of range for another node, it relays messages between the two, provided the relay node is in range for both.

1.4 Mobile App Requirements

To enable IOP testing framework on mobile, install Bluetooth mesh mobile app, version 3.1.0 or newer. The app is available for both Android and iOS.

1.4.1 Minimum Mobile Operating System Versions

The minimum OS versions supported by Bluetooth mesh mobile app are Android™ 9 and iOS®12.

2. Bringing up the Test Environment

The IOP test consists of a sequence of BLE operations executed between a mobile device and an EFR32 SoC running the interoperability test embedded software.

To flash the embedded software into one of the supported kits, plug the kit into the PC and open Simplicity Studio 5. In the Simplicity Studio launcher, select the “Example Projects & Demos” tab then filter by “Bluetooth mesh” technology type and disable the “Example Projects” toggle switch. This will narrow down the results to a few ready-to-flash demos, which include the IOP test, as shown below

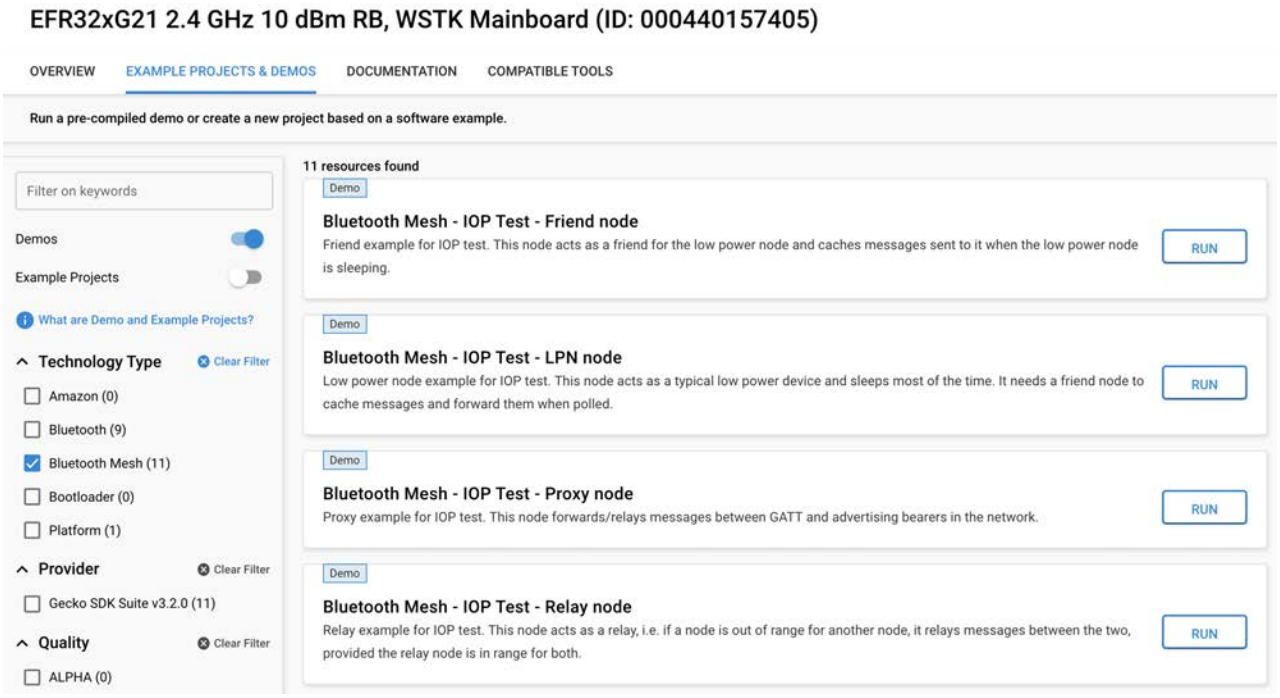


Figure 2.1. IOP Test Demo in Simplicity Studio 5

The embedded software can be flashed to the kit by clicking the “Run” button. After the flashing is complete, you should see the following information on the kit display, as shown below.

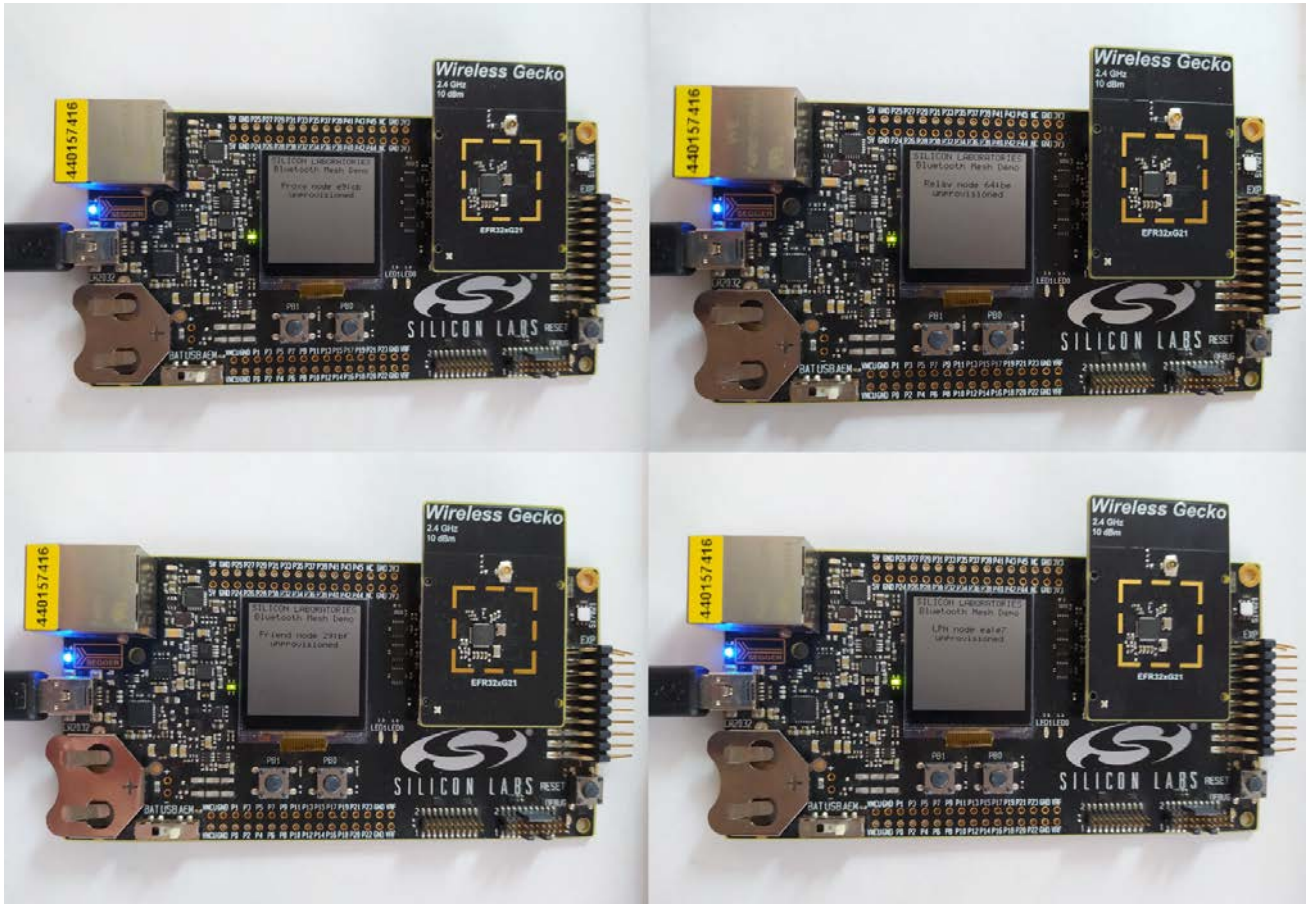


Figure 2.2. BRD4182A Running the IOP Embedded Software

For mobile, open Bluetooth mesh mobile app which will automatically open in the Networks view and tap on the IOP Test tab. Once you tap “Run Test” button, an alert dialog will appear. Interoperability Tests will begin after tapping “OK” button. Once you tap “Documentation” button, you will be redirected to the IOP execution instruction.

Note: It is recommended to change the screen sleep timeout to at least 5 minutes during testing. As stated in the warning in the app, if the screen turns off because of inactivity, tests will stop running.

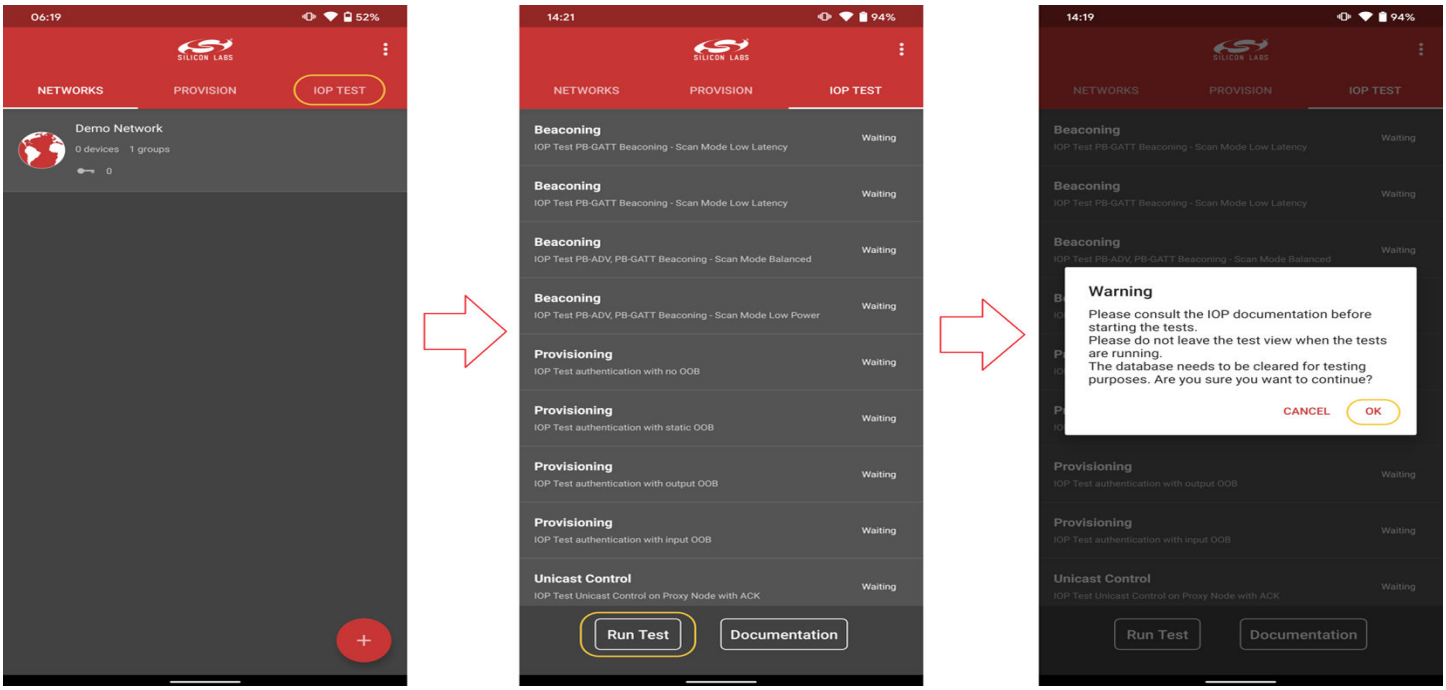


Figure 2.3. Launching the IOP Test on Bluetooth mesh Mobile App

3. Running the IOP Test

After the IOP test sequence starts running, the mobile app will scroll through the test cases and indicate Pass/Fail when the test is completed, as shown below.

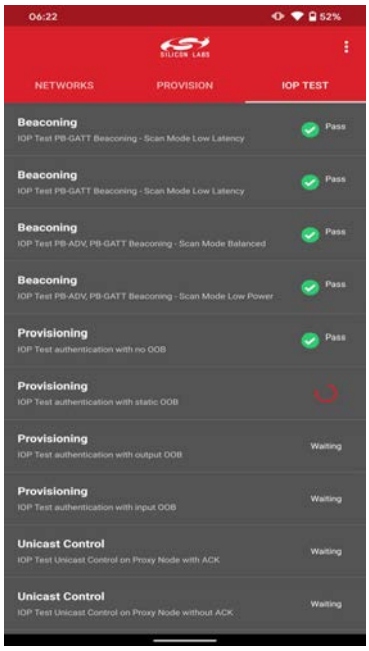
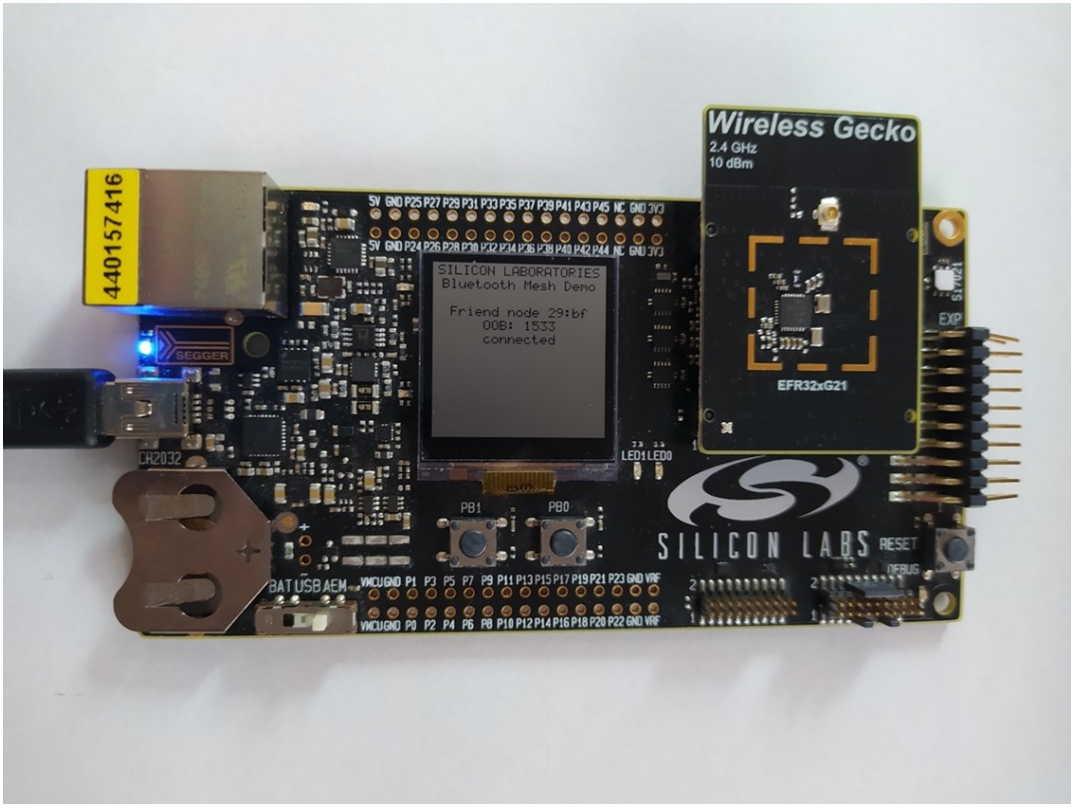


Figure 3.1. Ongoing IOP Test

Most tests do not require user intervention, the exceptions being: “IOP Test authentication with output/input OOB” and “IOP Test add a node to network”. To perform “IOP Test authentication with output OOB” or “IOP Test add a node to network”, enter an OOB data shown on node’s screen into an input field displayed in the app. Both node’s and app’s behaviors during these tests are shown below.



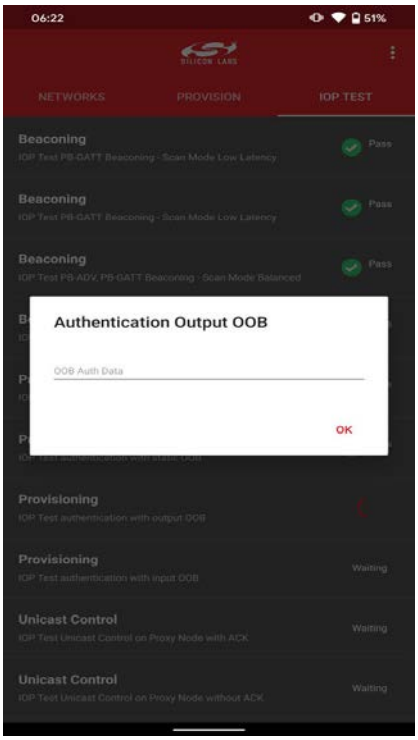
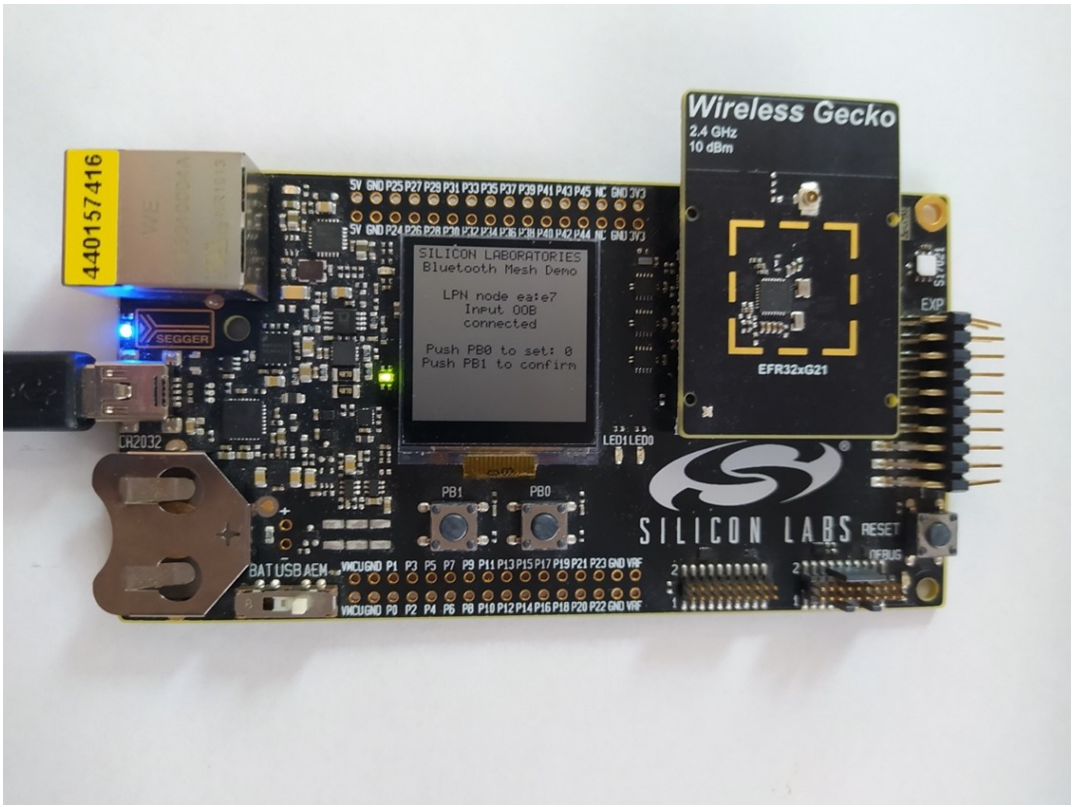


Figure 3.3. Authentication with Output OOB

As for “IOP Test authentication with input OOB”, you have to push board’s button “PB0” as many times as the app tells you to and confirm by pushing “PB1” button.



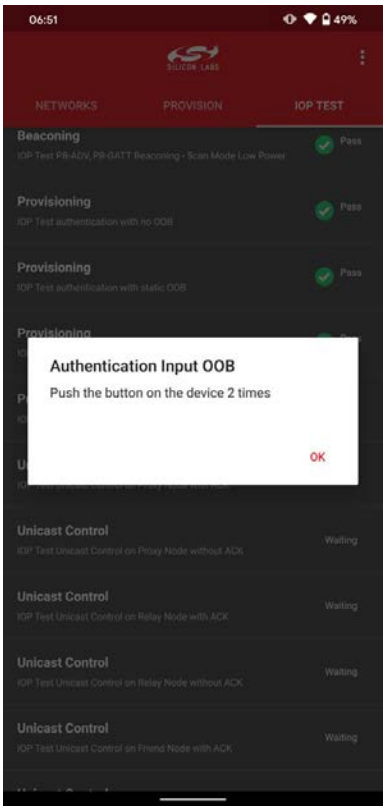


Figure 3.5. Authentication with Output OOB

4. Logging and Sharing Data

After the test is finalized on the mobile app, the user is given the option to rerun the test or share the results.

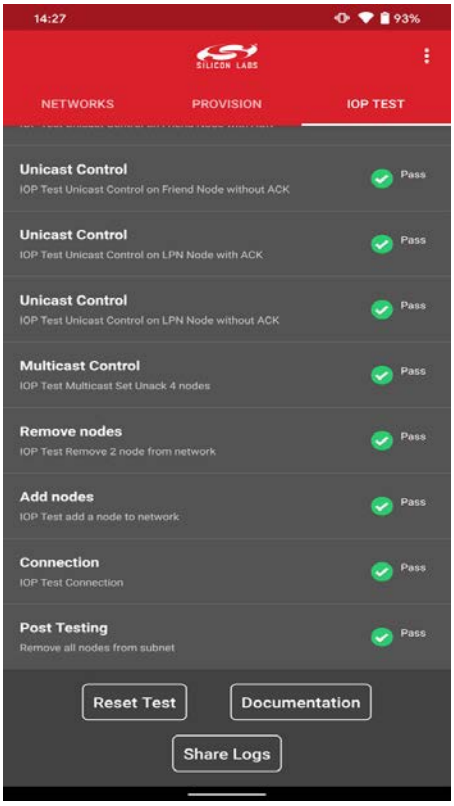


Figure 4.1. Run and Share Test Options

The Share option allows sharing the test log through OS-standard mediums such as cloud storage, email (iOS) or saving it locally (iOS and Android). The log is in .txt format and it contains information about the phone model, OS version and the result of each test. Below is an example of a test log from running IOP test on a Pixel 4 XL.

```
<timestamp>2021-06-08 07:17:11</timestamp>
<phone_informations>
        <phone_name>Google_Pixel_4_XL</phone_name>
        <phone_os_version>Android_SDK:_30_(11)</phone_os_version>
</phone_informations>
<firmware_informations>
        <proxy_node_uuid>00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F</proxy_node_uuid>
        <relay_node_uuid>00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 1F</relay_node_uuid>
        <friend_node_uuid>00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 2F</friend_node_uuid>
        <lpn_node_uuid>00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 3F</lpn_node_uuid>
</firmware_informations>
<test_results>
        Test case 1, PASS.Testing time: 218;Acceptable time: 1500.
        Test case 2, PASS.Testing time: 203;Acceptable time: 2500.
        Test case 3, PASS.Testing time: 3505;Acceptable time: 4000.
        Test case 4, PASS.Testing time: 2814;Acceptable time: 5500.
        Test case 5, PASS.Testing time: 2176;Acceptable time: 10000.
        Test case 6, PASS.Testing time: 2770;Acceptable time: 10000.
        Test case 7, PASS.Testing time: 9131;Acceptable time: 30000.
        Test case 8, PASS.Testing time: 4836;Acceptable time: 30000.
        Test case 9, PASS.
        Test case 10, PASS.Testing time: 9;Acceptable time: 180.
        Test case 11, PASS.
        Test case 12, PASS.Testing time: 7;Acceptable time: 180.
        Test case 13, PASS.
        Test case 14, PASS.Testing time: 12;Acceptable time: 180.
        Test case 15, PASS.
        Test case 16, PASS.Testing time: 7;Acceptable time: 180.
        Test case 17, PASS.Testing time: 6;Acceptable time: 180.
        Test case 18, PASS.
        Test case 19, PASS.
        Test case 20, PASS.Testing time: 2156;Acceptable time: 30000.
        Test case 21, PASS.
</test_results>
```

Figure 4.2. IOP Test Log Example

4.1 Collecting Additional Data from the Embedded Device

The IOP embedded software also sends logging data over UART, which can be captured by a terminal emulator on the PC. Furthermore, the Packet Trace Interface (PTI) is enabled which means that the radio traffic can be captured using the Network Analyzer.

For a more comprehensive data set around an individual IOP test sequence, capture both UART logs and radio traces using Simplicity Studio. Radio trace capture must be initiated before starting the IOP test.

To start the radio capture, right click on the debug adapter and select "Connect". Then, right click once again and select "Start Capture".

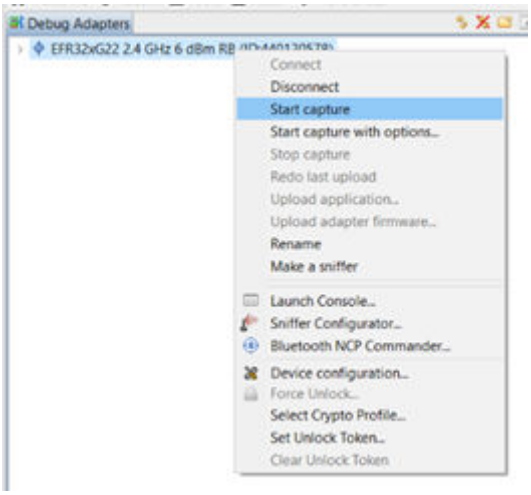


Figure 4.3. Initiate Radio Traffic Capture Using Network Analyzer

This will automatically bring up the Network Analyzer view where the traffic is logged and every packet can be decoded for further analysis if required. At the end of the IOP test, the trace can be saved through File -> Save as, as shown below.

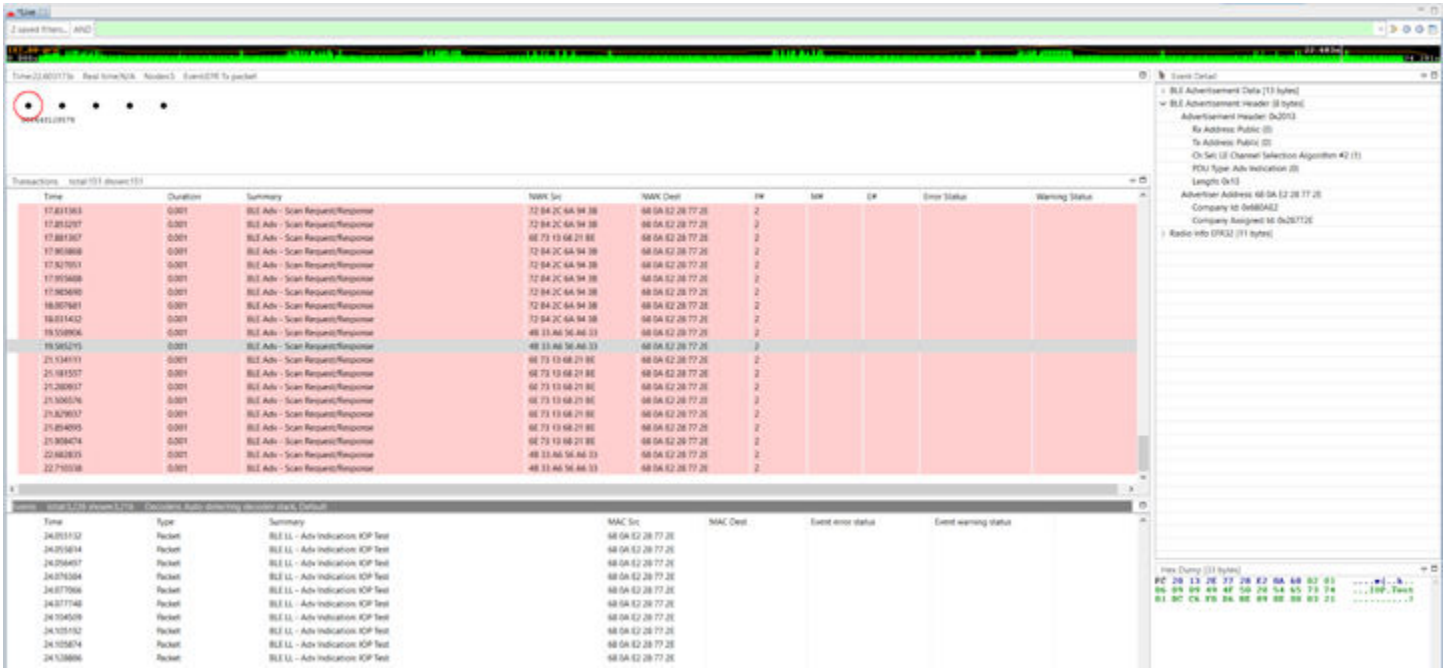


Figure 4.4. Simplicity Studio Network Analyzer

While UART logs can be captured by COMPort emulators such as tera term, you can also capture within Simplicity Studio's own terminal. Similarly to the packet trace capture, right click on the debug adapter and select "Launch Console". Then, select "Serial 1" tab and enter any character into the prompt at the bottom of the screen. You should see the connection symbol changing, which indicates that the connection is successful and the logging should start, depending at which phase of the IOP test this is initialized. Otherwise, reset the board to ensure that the log is being received.

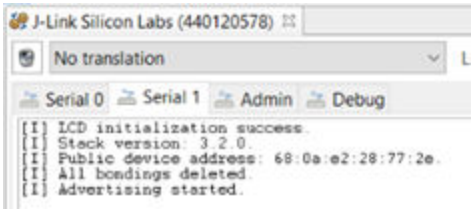


Figure 4.5. Simplicity Studio Console

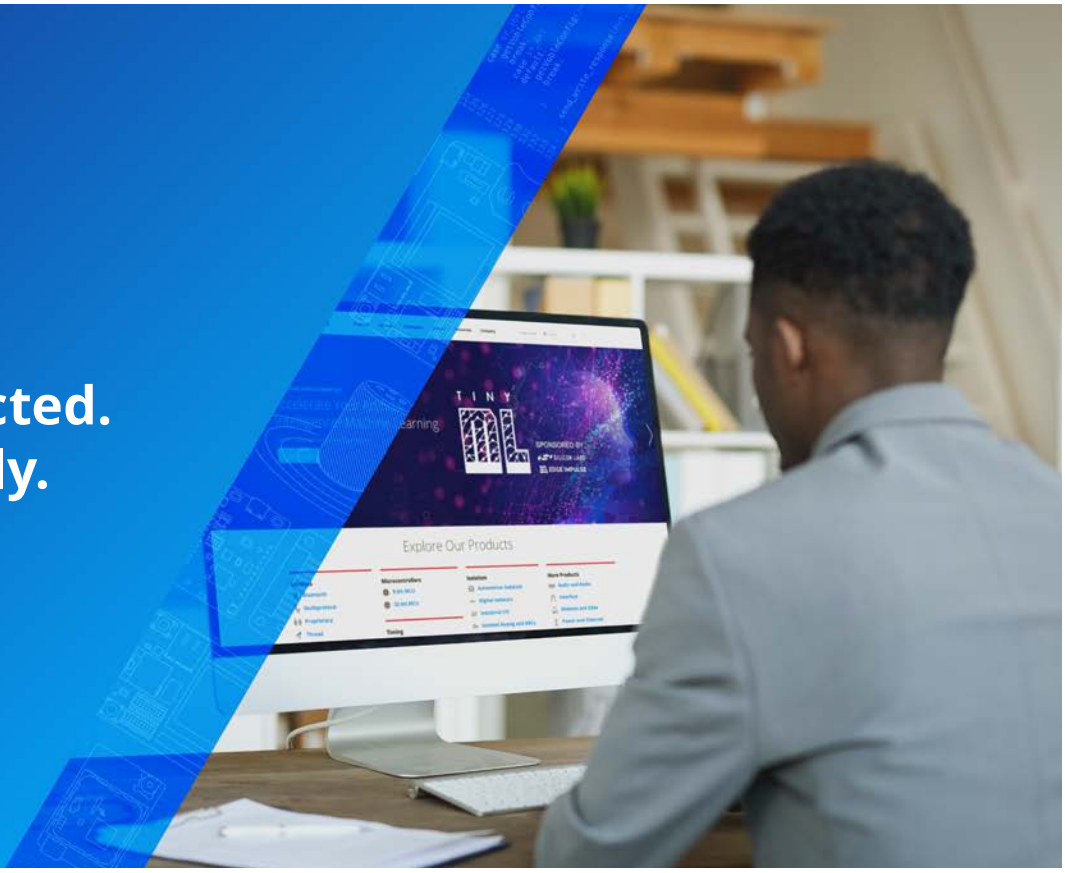
5. Revision History

Revision 0.1

May, 2021

- Initial release

Smart. Connected. Energy-Friendly.



IoT Portfolio
www.silabs.com/products

Quality
www.silabs.com/quality

Support & Community
www.silabs.com/community

SILICON LABS

Silicon Laboratories Inc.
400 West Cesar Chavez
Austin, TX 78701
USA

www.silabs.com