AN1317: Using Network Analyzer with *Bluetooth*® Low Energy and Mesh

This document briefly describes the features of the Simplicity Studio 5 Network Analyzer for Bluetooth Low Energy and Mesh.

It can be used and read jointly with the *AN958: Debugging and Programming Interfaces for Customer Designs* for more information on PTI usage with custom hardware.

This document assumes familiarity with the basic Network Analyzer information in the Simplicity Studio 5 User's Guide.

KEY POINTS

- Packet Trace Interface
- Wireless Network Analysis
- ISD capture files
- Bluetooth mesh networks
- Bluetooth mesh security keys
- Bluetooth Low Energy

Table of Contents

1 Introduction

Network Analyzer is a tool for analyzing wireless network traffic. It supports a wide variety of short-range wireless protocols like Bluetooth Low Energy, Zigbee, proprietary protocols and others. It is provided as part of the Simplicity Studio tool set.

1.1 Debugging a Wireless Network

Silicon Labs' tool set provides the user with a comprehensive way to analyze wireless traffic. With it, the user can tap into the data buffers of the radio transceiver via a dedicated serial hardware interface called the Packet Trace Interface (PTI). PTI data can be then transferred via USB or Ethernet to a computer running Simplicity Studio. Finally, the time-stamped data can be interpreted and displayed in Network Analyzer.

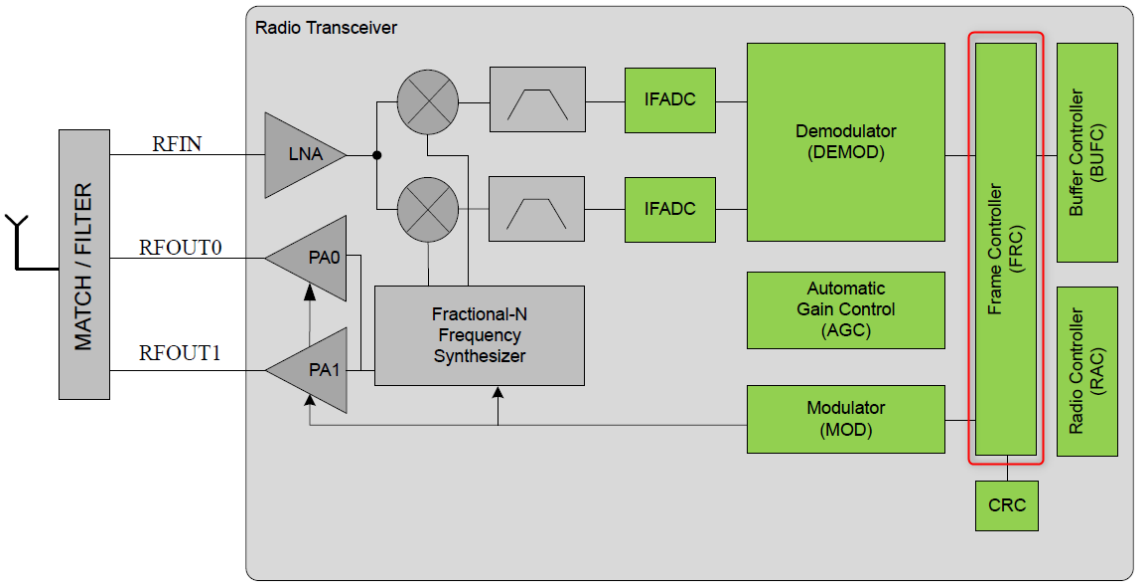
Most Silicon Labs' development kits, such as the Wireless Starter Kit (WSTK), have the PTI embedded and ready to use. Note that it is also possible to use the network analysis features when working on custom hardware if the PTI pins are exposed via a debug interface.

2 Hardware and Packet Trace Interface (PTI)

On the EFR32 series 1 and 2, a mechanism is provided for the user to be able to tap into the data buffers at the radio transmitter/receiver level. From the embedded software perspective, this is available through the **RAIL Utility, PTI** component in Simplicity Studio. That component is effectively a simple packet trace interface driver.

2.1 Packet Trace Interface

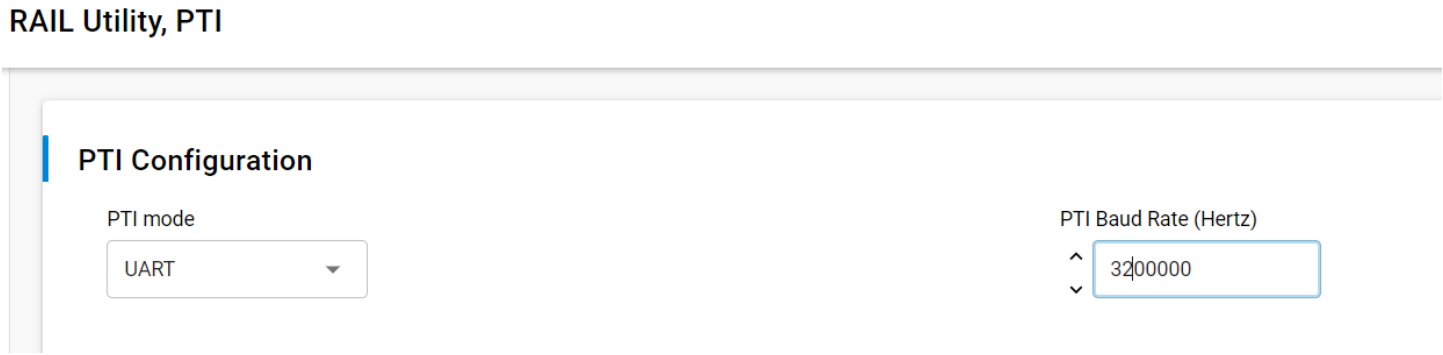
The Packet Trace Interface (PTI) is an interface giving serial data access directly to the radio transmitter/receiver frame controller. The following figure describes at a high level the architecture of the radio transceiver.



A clock and data signal are connected to the frame controller to monitor all packets received/transmitted by the chip. At the chip level, a signal is dedicated to trigger the timestamping of each PTI frame by the WSTK board controller. The PTI is a non-intrusive sniffer of data, radio state and time stamp information.

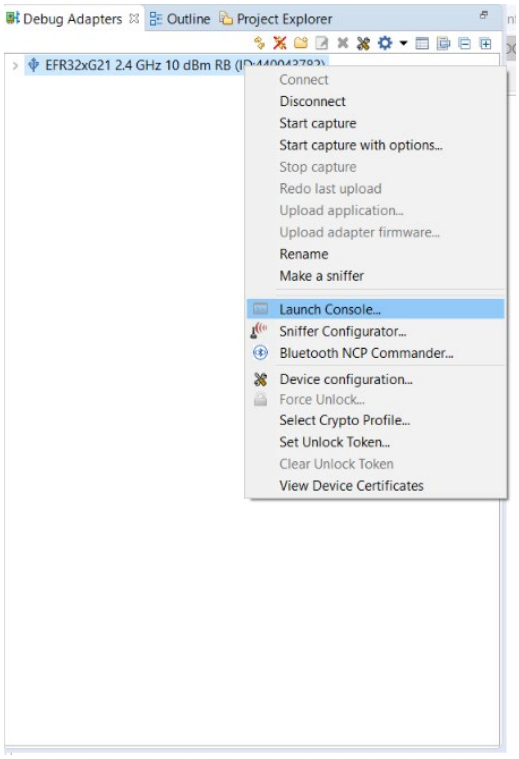
A single-pin UART signal is used for PTI data transfer. This can be configured in the **RAIL Utility, PTI** component. The baud rate is selectable. The default baud rate is 1.6 Mbps. The maximum baud rate is 3.2 Mbps.

When using 2M PHY with Bluetooth Low Energy, the default PTI-over-UART speed (1.6 Mbps) must be increased to a higher baud rate. The following shows an example:



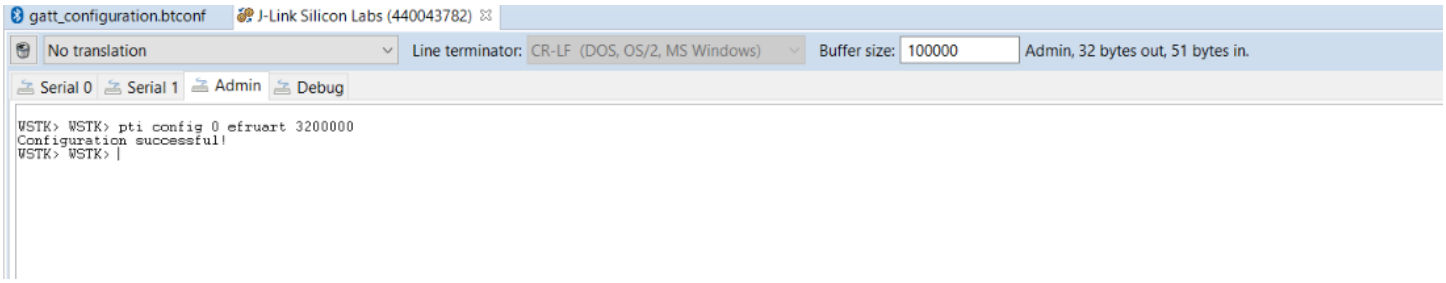
Additionally, the speed at which the PTI frames are forwarded from the EFR32 back to USB/UART must also be increased. This is done by setting the PTI config corresponding to your adapter at the correct baud rate through the Admin Console interface.

In the Debug Adapters view, right-click the device. In the context menu, select **Launch Console…**



Select the Admin tab, and execute the command:

```
pti config 0 efruart 3200000
```



2.2 Hardware Kit and PTI

As described previously, the WSTK can be used to monitor the wireless traffic. The WSTK can be connected to the PC via USB or Ethernet:

- USB - The simplest solution. This is used throughout this document.
- Ethernet - Recommended for best performance and scalability.

Figure 2.1. WSTK Board

Alternatively, it is possible to use the PTI on custom hardware if the corresponding pins are exposed (via the 20-pin Simplicity Connector on the WSTK or the Simplicity Mini connector on the debug adapter for example):

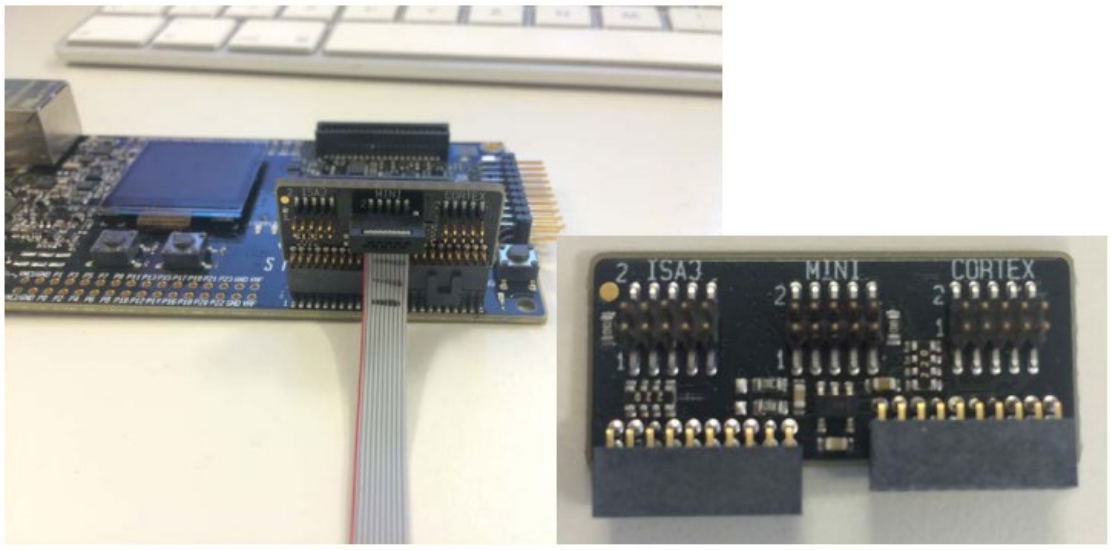


Figure 2.2. Simplicity Debug Adapter Exposing PTI Signals on the Mini and Simplicity Connectors

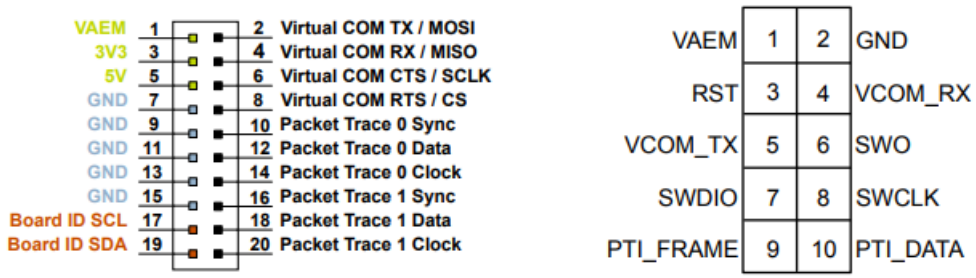


Figure 2.3. 20-Pin Simplicity Connector and the Mini Connector

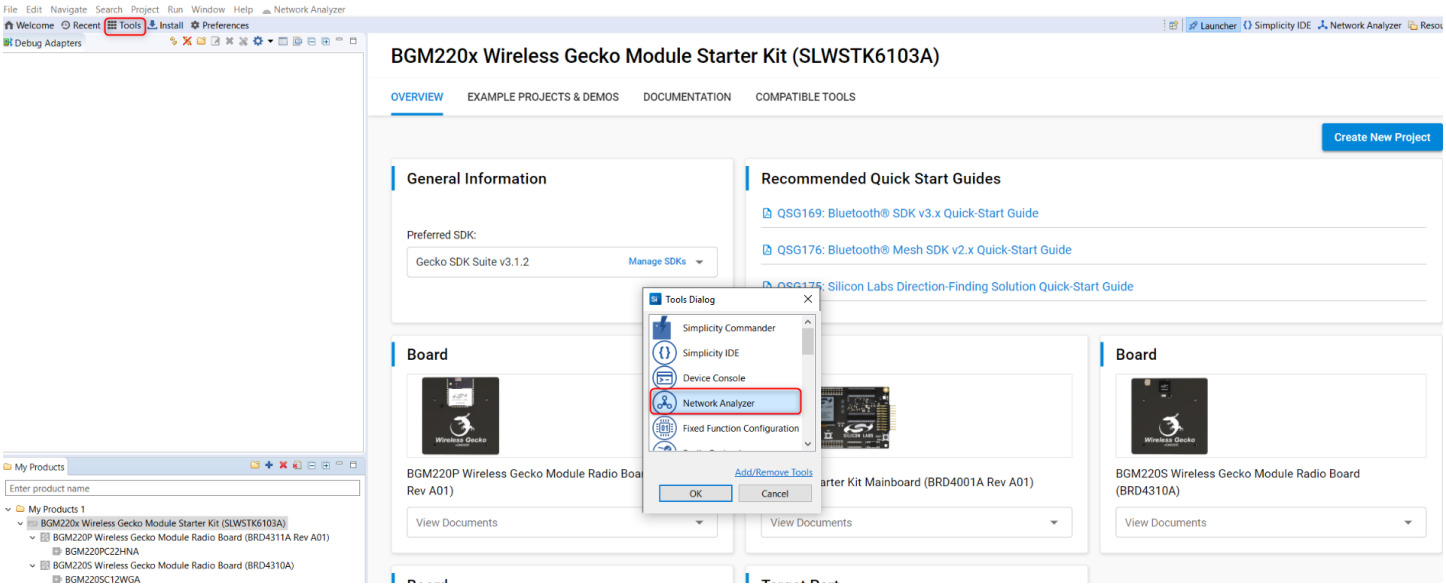
For more detail, refer to *AN958: Debugging and Programming Interfaces*.

3 Network Analyzer Features

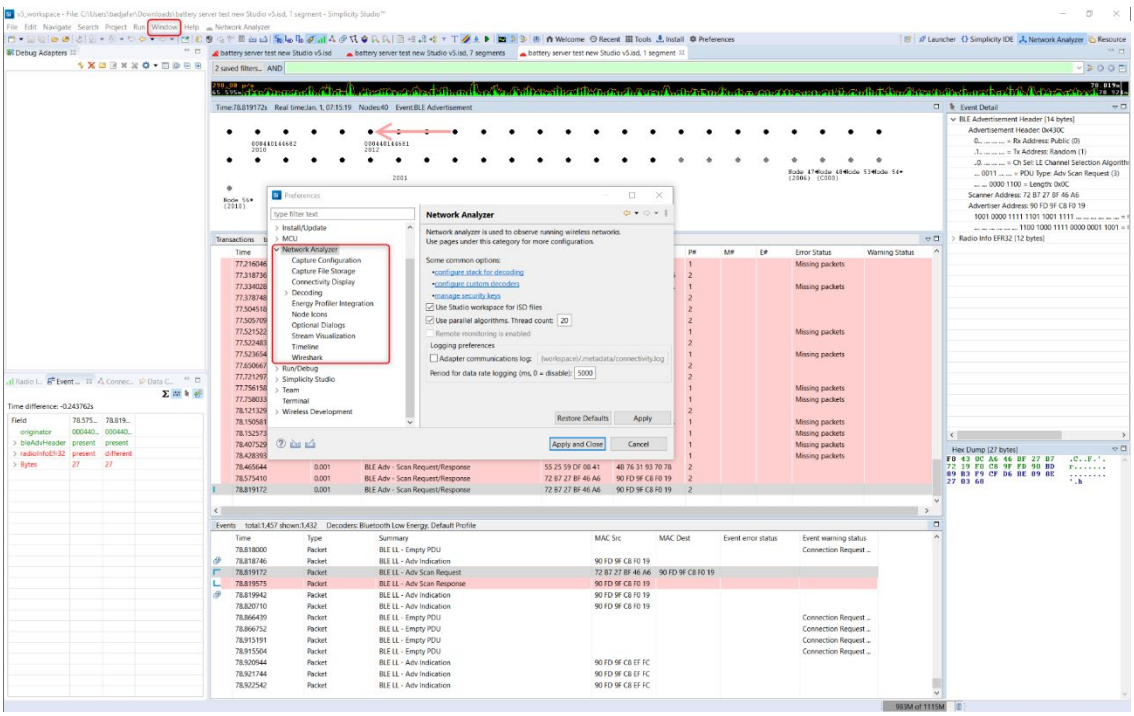
This section describes the main features of the Network Analyzer user interface.

3.1 Tool Access and Preference Page

Network Analyzer is provided with Simplicity Studio. For easy access, on the toolbar click **Tools**, and select **Network Analyzer**.



With Network Analyzer Preferences, accessed by **Windows > Preferences** or the **Preferences** control on the toolbar, you can tune the tool, add security keys, and modify displays and icons.

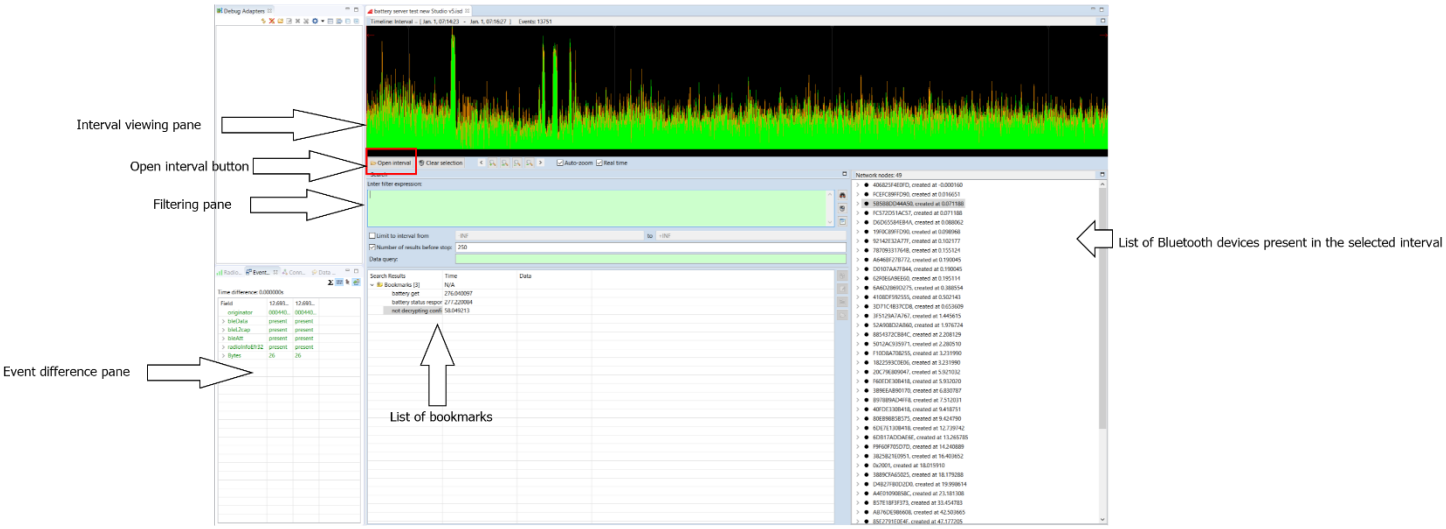


You can also modify the Bluetooth protocol decoder in great detail (under the Decoding menu), define the integration with other tools (Energy profiler, Wireshark), and add Security keys (see section 4.2.2 Keys for more information).

3.2 Interface

3.2.1 Large File Editor

Because radio traffic contained in a captured session can be very large, a pane allowing time interval selection is opened first, as shown in the following figure. The interface implementing that feature is called the Large File Editor. It gives a complete overview of the capture session and allows you to select a specific time interval of study.



Interval viewing pane: Displays the traffic data present in the ISD live or recorded session. The data points corresponding to the green curve represent the number of packets per unit of time. Further display options are available through the right-click context menu.

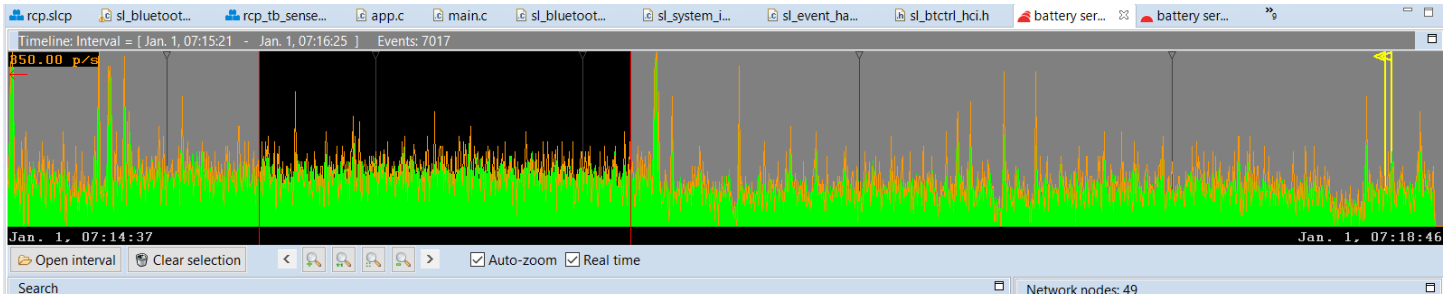
Open Interval button: Once you have selected the area of interest in the interval view, click **Open Interval** to see traffic detail. This opens a subsequent window in which transaction and events are listed, as shown in section 3.2.2 Interval Editor.

List of nodes: In the right pane, the list of nodes present in the selected time interval is shown.

Filtering pane: Some packet filtering can be done on the selected time interval. In practice, this is more useful in the transaction/event pane. Filters are explained in more detail in section 3.6 Filters.

Bookmarks and Event Difference pane: Those functions are explained in detail in sections 3.4 Bookmarks and 4.1 Network Analyzer for Bluetooth LE, respectively

The following figure illustrates a selected interval (using the mouse directly on the waveform). The interval of interest appears in clear, whereas the rest of the interval viewing pane is greyed out.



3.2.2 Interval Editor

The Editor is laid out as follows:

The events displayed in the event pane are directly linked to the transactions displayed in the transaction pane. When a transaction is selected, Network Analyzer jumps to the corresponding events in the event pane.

Press CTRL+SHIFT+Up/Down to toggle between events in a transaction.

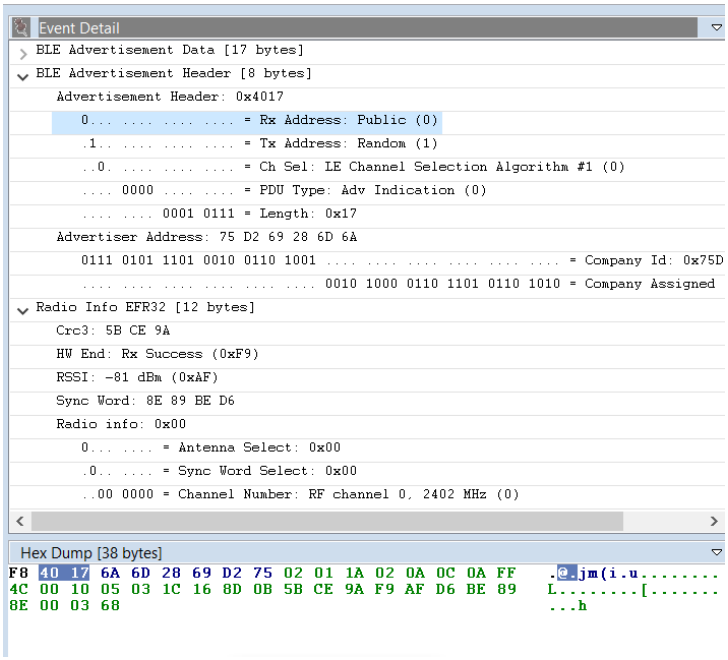
In the **Status** field, “Missing packets” indicates that a packet expected in the transaction has not been received within the current timeout window.

Note: A transaction is a group of related packets that together form a higher layer protocol procedure or message exchange. In the case of Bluetooth Low Energy (and Bluetooth mesh), a “transaction” refers to an actual Bluetooth Low Energy transaction as defined in the core specification. This corresponds most of the time to a Bluetooth Low Energy procedure. Equally, the event pane displays the actual Bluetooth Low Energy events corresponding to the transaction or procedure. For more details, refer to the Bluetooth Core specification document.

The filtering capabilities of the tool are explained in detail in section 3.6 Filters.

Event detail pane: The event detail pane exhibits all the data present in the corresponding Bluetooth LE packet. The data are displayed in a decoded format allowing the user to unpack the information at all layer levels, from radio information up to the GATT protocol and

Bluetooth mesh Access layer. Using the Option menu in the top right corner of the Event Detail pane, you can expand the bit fields and toggle between hexadecimal and decimal format.



Note that when a bit field is selected, the corresponding data is automatically highlighted in the hex dump view of the packet.

Various layers of decoding exist, which can be viewed in the Hex Dump Pane (for Bluetooth mesh, result of Network decryption, Application decryption, Defragmentation of various levels and so on).

3.3 ISD File

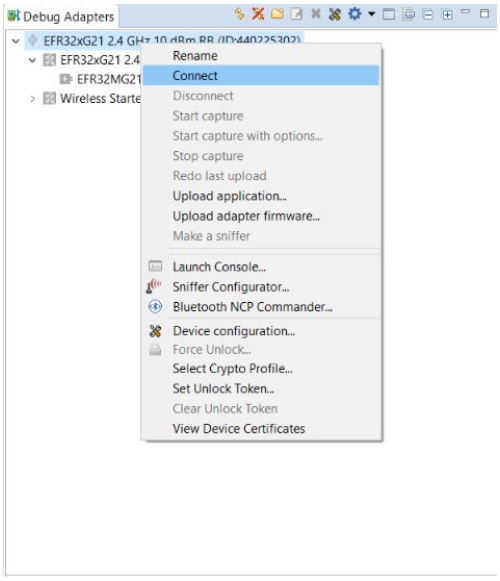
Network Analyzer can capture data from nodes of any connected adapters, either from one node at a time or from multiple nodes. It can display data from Live sessions as well as Recorded sessions.

Network Analyzer saves session data to an ISD (.isd) file, which is a compressed file that stores session data and the network state. Network state includes display settings such as map modifications, which Network Analyzer restores when you reload the session file.

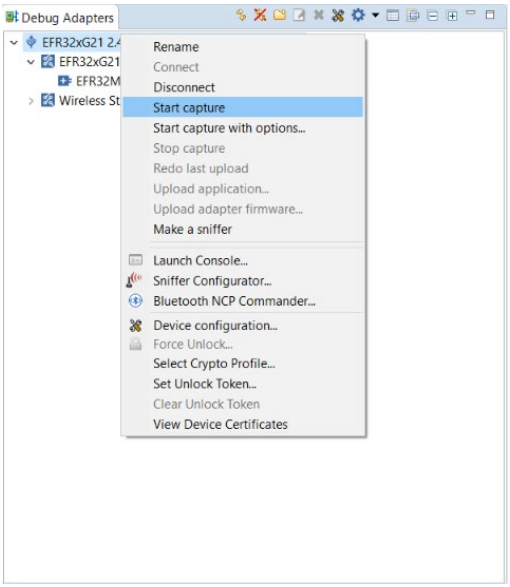
The following procedure describes how to start a network data capture on a single device:

1. In **Preferences > Simplicity Studio > SDKs** select the desired SDK.
2. In **Preferences** > **Network Analyzer** > **Decoding** >**Stack Version,** Make sure “Bluetooth Low Energy” is added in the decoding preferences.
3. If using encryption with known keys, make sure the security keys are added (see section 4.2.2 Keys for more detail).

4. Connect to the adapter.



5. Start capture.

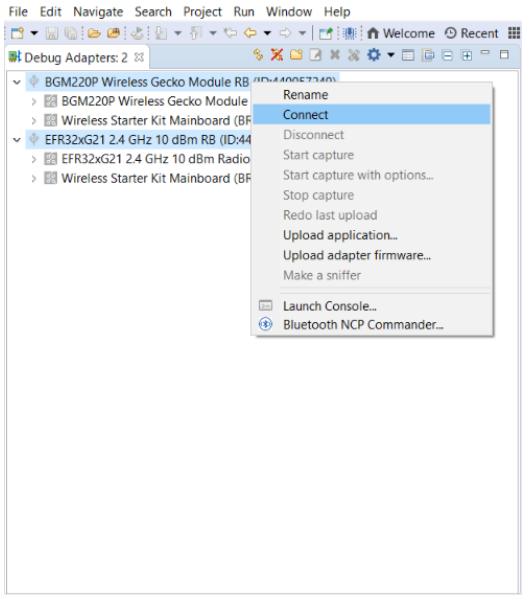


To start capturing on several adapters, press and hold the CTRL key and, in the Debug Adapters view:

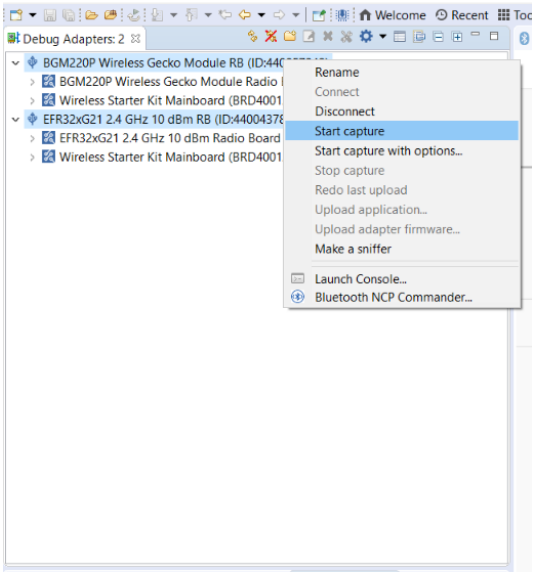
1. Select more than one adapter.
2. Right-click and select **Connect**.

3. Right-click and select **Start Capture.**
4. Release the CTRL key.

a)



b)



a) Connect to a Group of Adapters. b) Start Capture on a Group of Adapters

With Bluetooth mesh, when working in a busy environment, capturing from multiple nodes makes more sense than capturing from only one, because Bluetooth mesh nodes are constantly scanning for incoming packets.

For example, some nodes might be emitting Bluetooth mesh messages or beacons at the same time the node being used for capturing is answering a configure request. As a result, the data that were sent will not be received, simply because the link layer of the underlying Bluetooth LE stack was in the advertising state and not scanning. Additionally, there are three primary advertising channels and the radio transceiver can only listen to one channel at a time.

This should not be mistaken for an error or a malfunction in the stack. Rather, this is simply a by-product of the technology reliance on primary advertising channels and advertising PDUs. Apart from the GATT bearer scenario (relying on Bluetooth LE connections), there is no communication collision control.

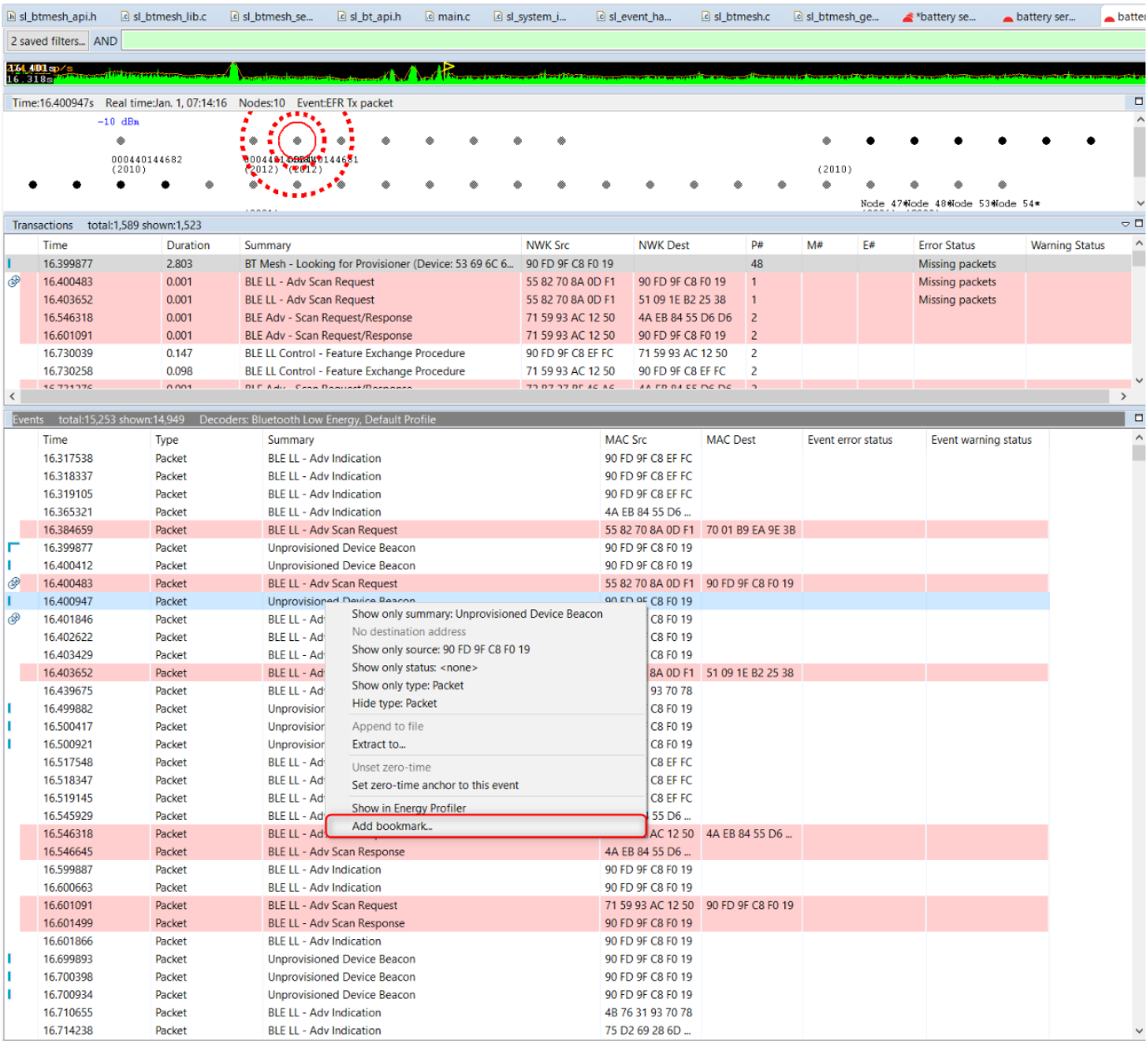
Note: It is possible to use “Duplicate Detection”, i.e. when the same packet is detected on multiple adapters it will only be displayed once. See **Preferences->Network Analyzer->Capture Configuration** for more details.

3.4 Bookmarks

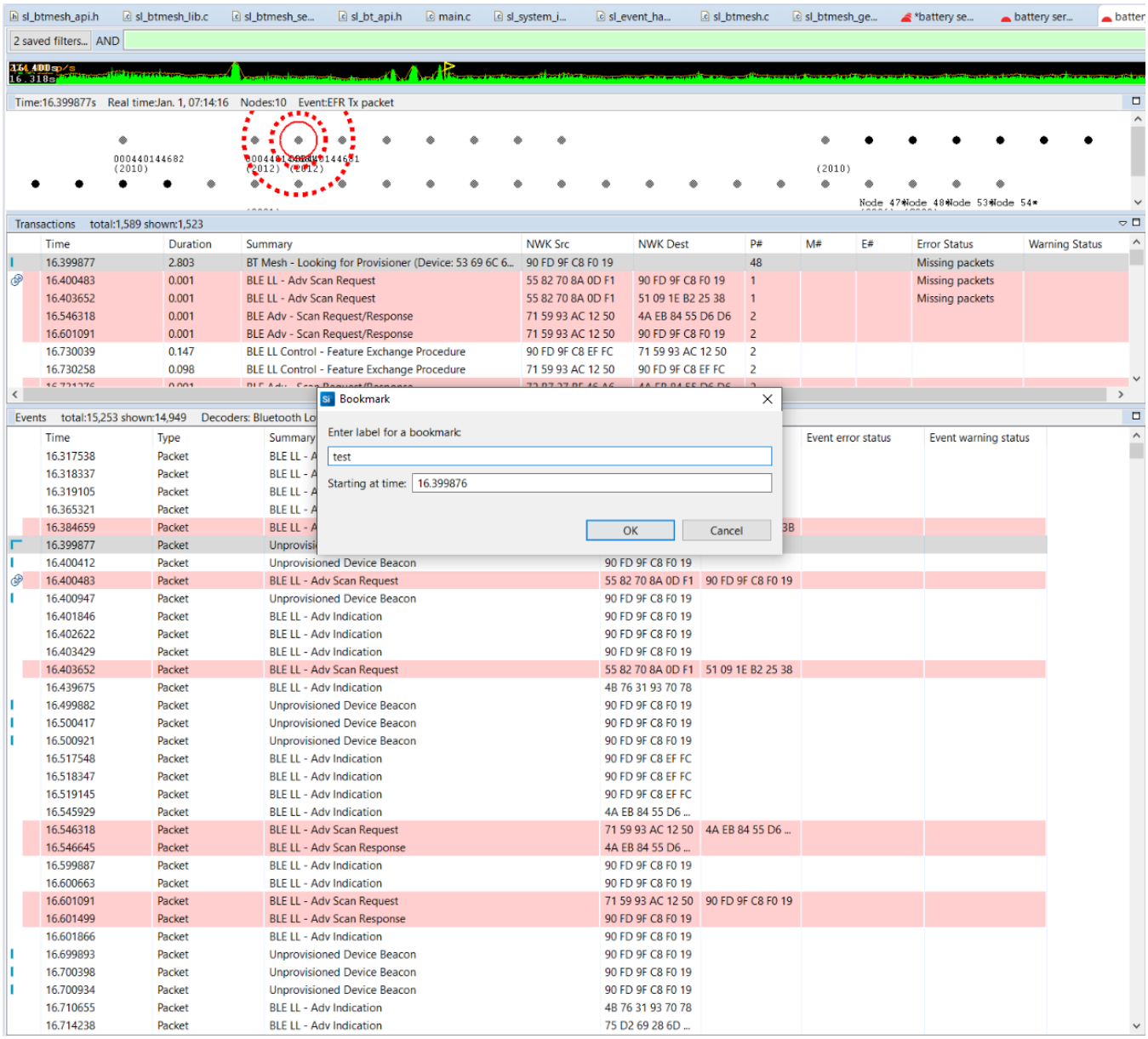
Bookmarks can be used for marking events. As the name indicates, those are actual bookmarks that point to a certain Bluetooth LE or mesh event. Those are useful for pinpointing a certain event in a transaction that can be problematic and then sharing it with somebody else.

The following procedure shows how to set and navigate through bookmarks:

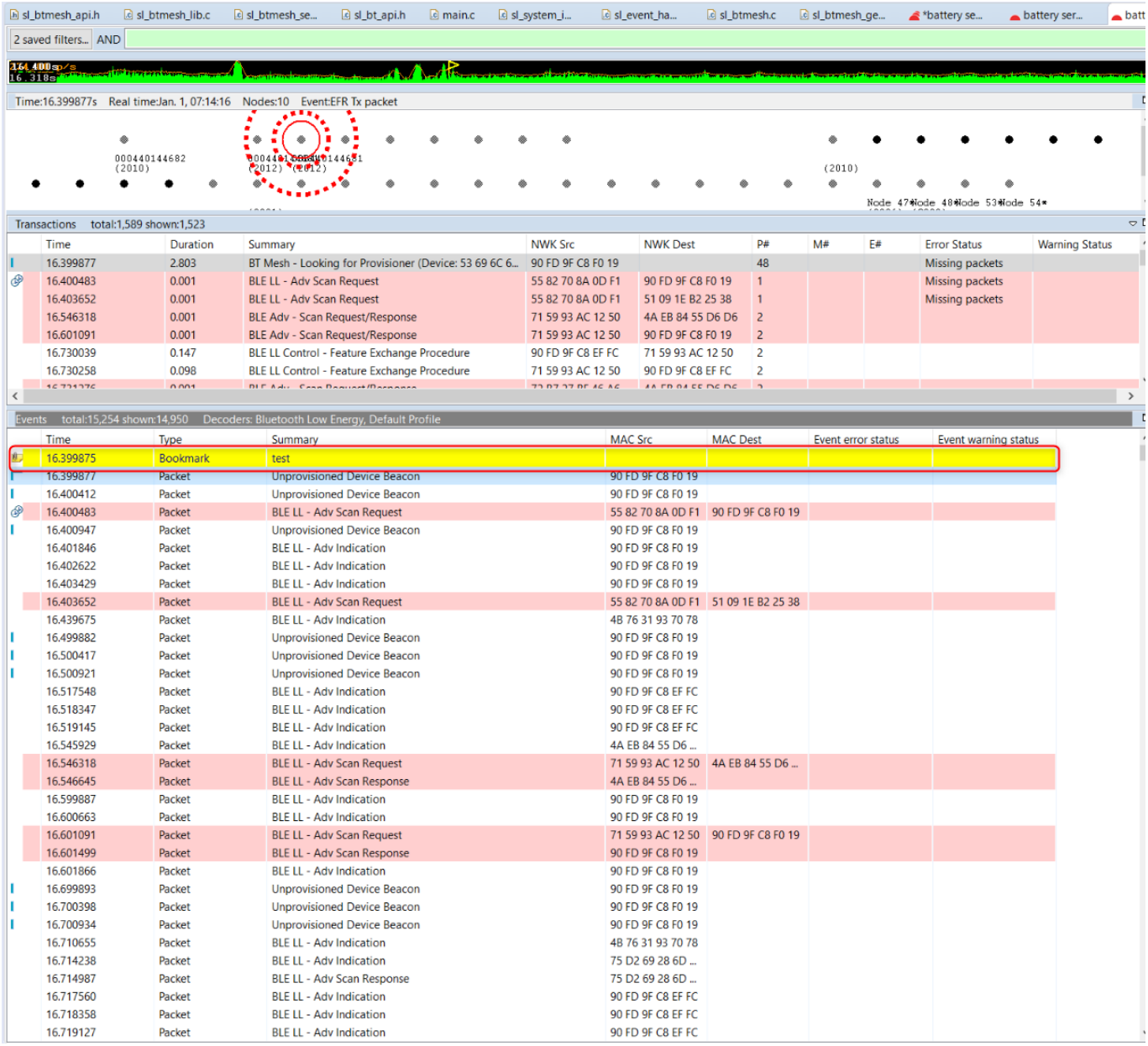
1. Select the event and right-click it.



2. Enter the bookmark name.



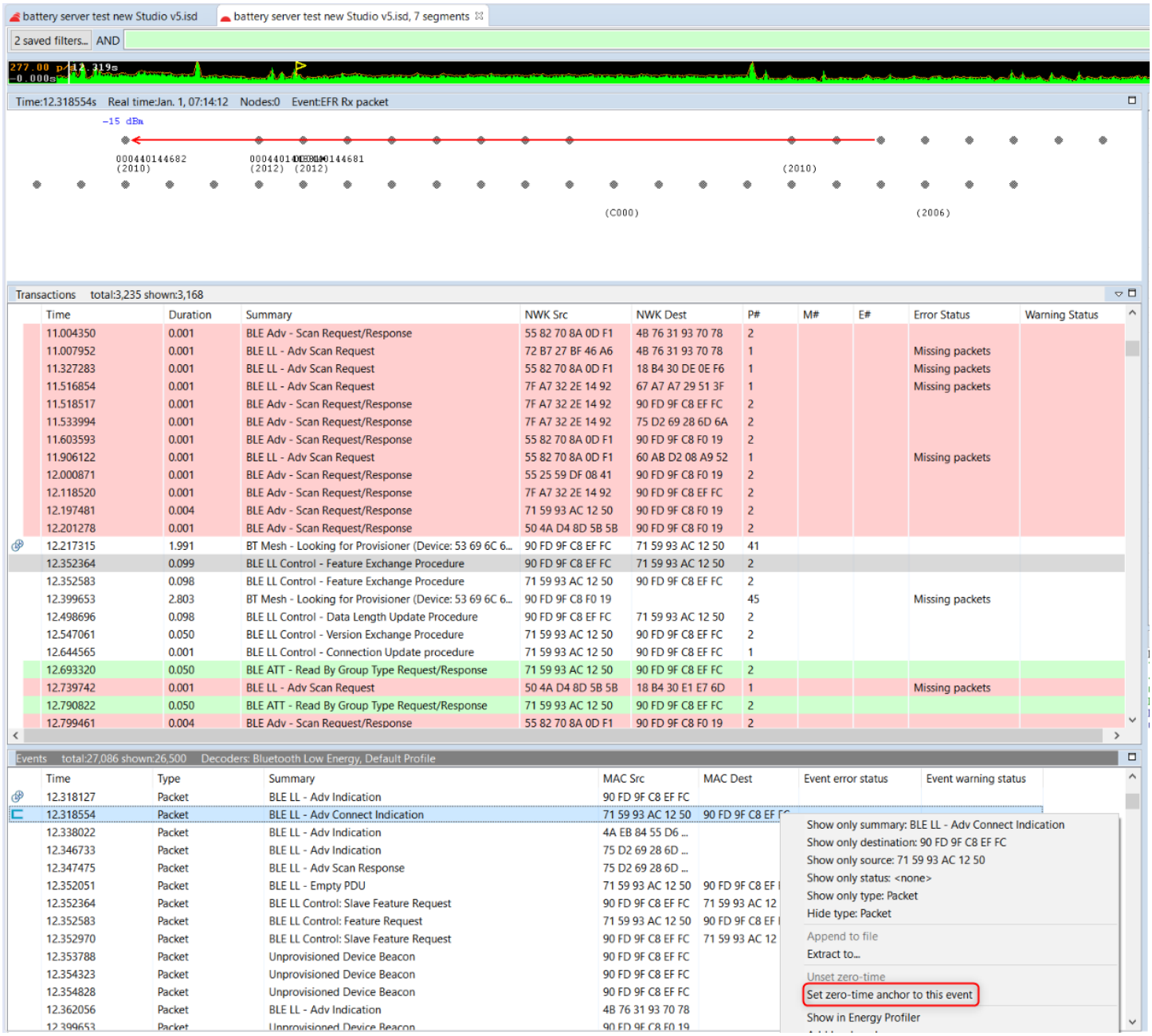
3. See the bookmark recorded (highlighted in yellow).



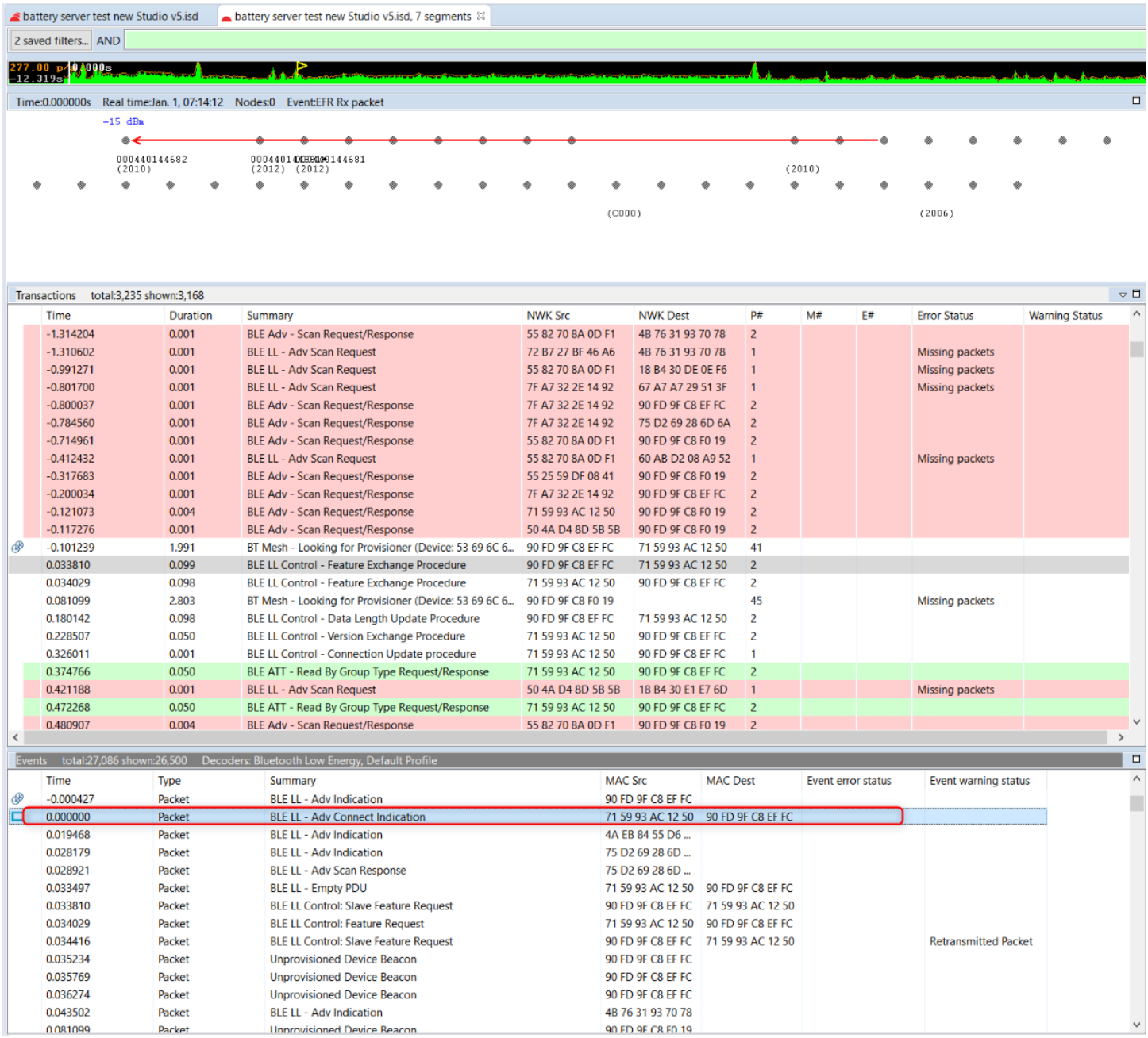
3.5 Set Zero-Time Anchor

When studying a particular event or transaction, it is sometime useful to set it as the time reference. Practically, this means setting the timestamp corresponding to that event or transaction to zero, and then seeing all subsequent timestamps updated according to the new time reference. Using this Network Analyzer feature is also an excellent way to verify the Bluetooth Low Energy advertising or connection timings (advertising interval, connection interval, and so on).

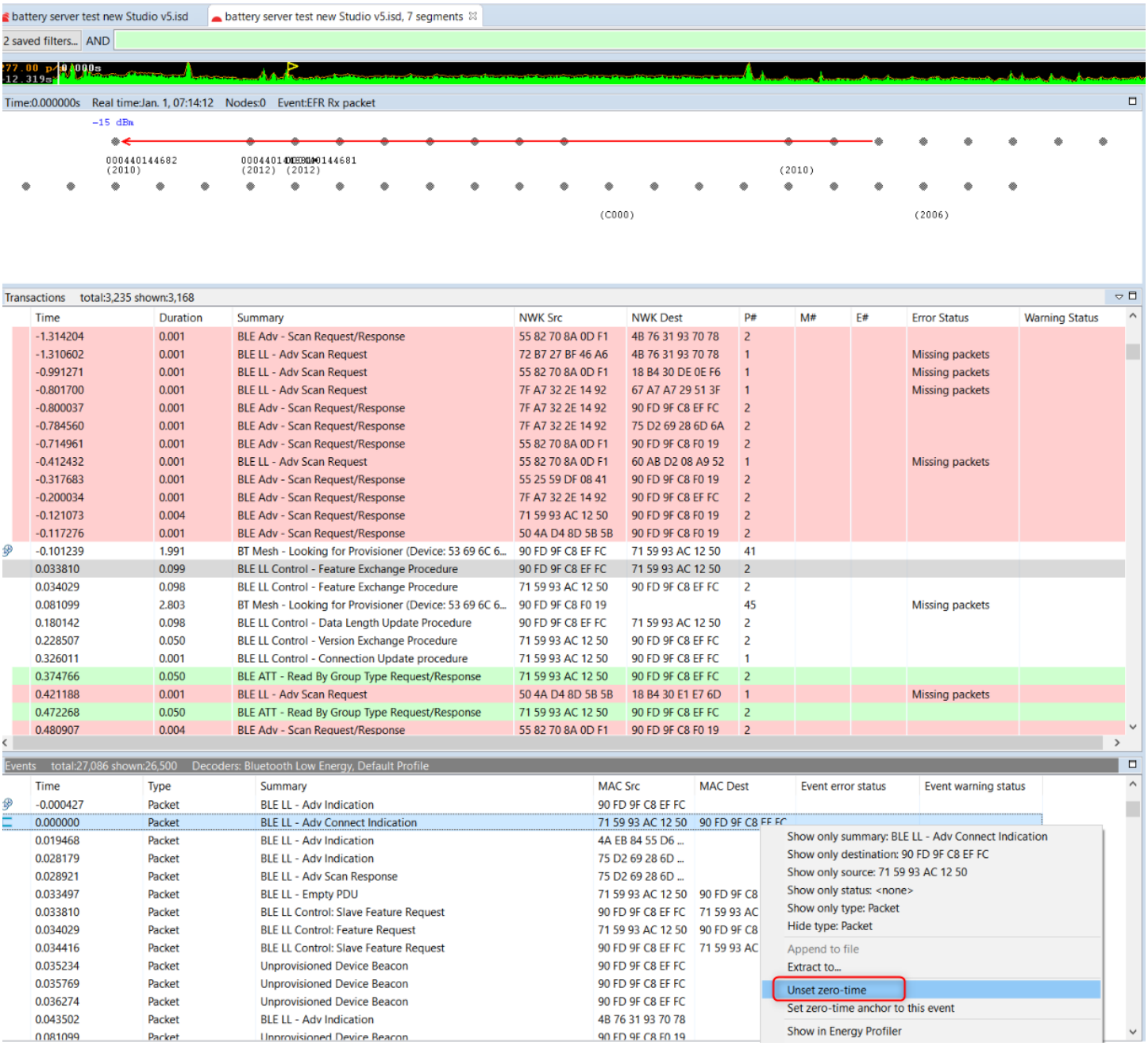
The following describes how this can be done, using a Bluetooth LE Initiating connection (CONNECT_IND) state as an example. Select the particular transaction or event, right-click to open the context menu, and then click **Set zero-time event anchor to this event.**



The timestamps of all events and transactions are then updated, taking into account the new anchor as time reference, as shown in the following figure.



To remove the anchor, right-click the selected event or transaction and click **Unset zero-time**.

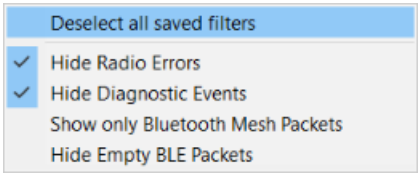


3.6 Filters

Network Analyzer supports use of a set of built-in and manual filters.

3.6.1 Built-In Filters

The following built-in filters can be enable/disabled when visualizing PTI data:



For example, you can filter Bluetooth LE data out in order to focus on Bluetooth mesh traffic. Radio errors and diagnostics can also be filtered out. Note that radio errors can be useful for debugging.

3.6.2 Manual Filters

The filter bar is used to filter transactions and events. Each capture session has its own filters settings. When a session's filter is changed and the filter is applied by clicking **Apply** in the Filter pane toolbar, Network Analyzer will refresh the display showing only the corresponding transactions or events. When you exit Network Analyzer, all sessions filters are cleared and must be reapplied when Network Analyzer is restarted.

Network Analyzer provides two ways to edit filters:

- Filter Manager: Maintains internally a set of saved filters that you can review and edit. You can also add new filters. You specify any of the saved filters for display on the Filters menu, where they are accessible for use in one or more sessions.
- Filter Bar: An editor that attaches to a given session, where you can enter one or more filter expressions on the fly. Network Analyzer discards filter bar expressions for all sessions when it exits. It does, however, store the expression for easier future access.

Multiple filters can be combined using logical expressions:

- `&&` - And operator
- `||` – Or operator

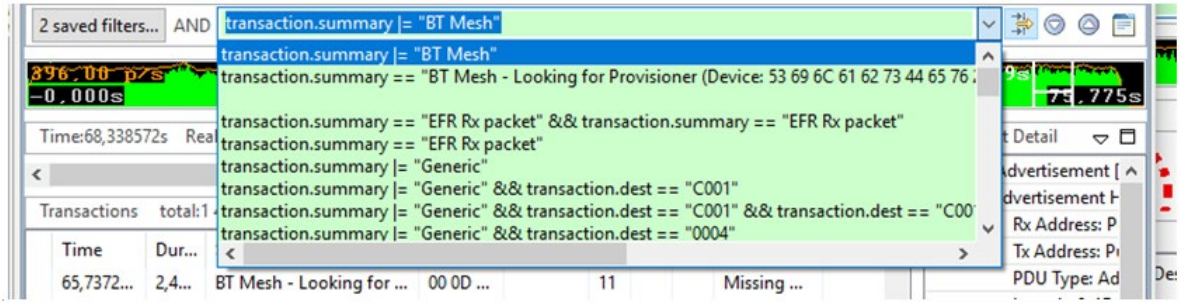
Alternatively, conditions for individual filters can also be used:

- `==` - Equals
- `!=` - Not equal
- `|=` - Contains

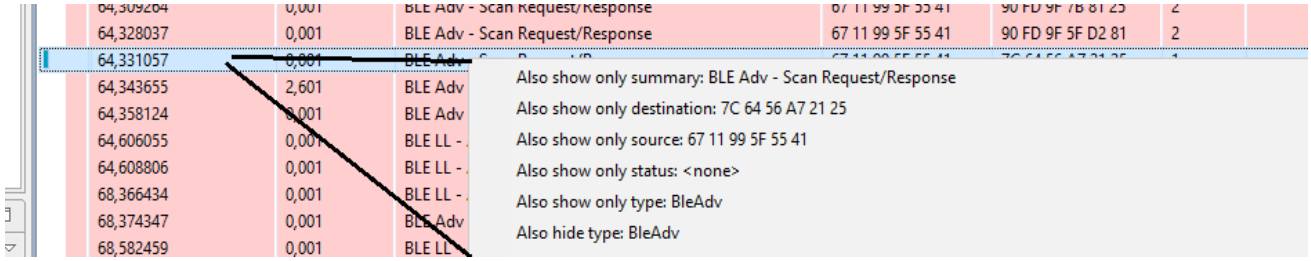
The following table shows examples of filtering in Bluetooth mesh traffic:

Filter example	Meaning
transaction.summary |= "BT Mesh"	Show transactions where the **summary** field contains the text ”BT Mesh”
transaction.summary |= "Generic" && transaction.dest == "C001"	Summary contains string **Generic** and the destination address is 0xC001
transaction.summary != "EFR Rx packet"	Do not show transactions with summary ”EFR Rx packet” -> Hide those transactions that Network Analyzer cannot decode

The following figure shows an example of saved combined filters.



Alternatively, you can right-click and select preset filters.



4 Network Analyzer for Bluetooth LE and Mesh

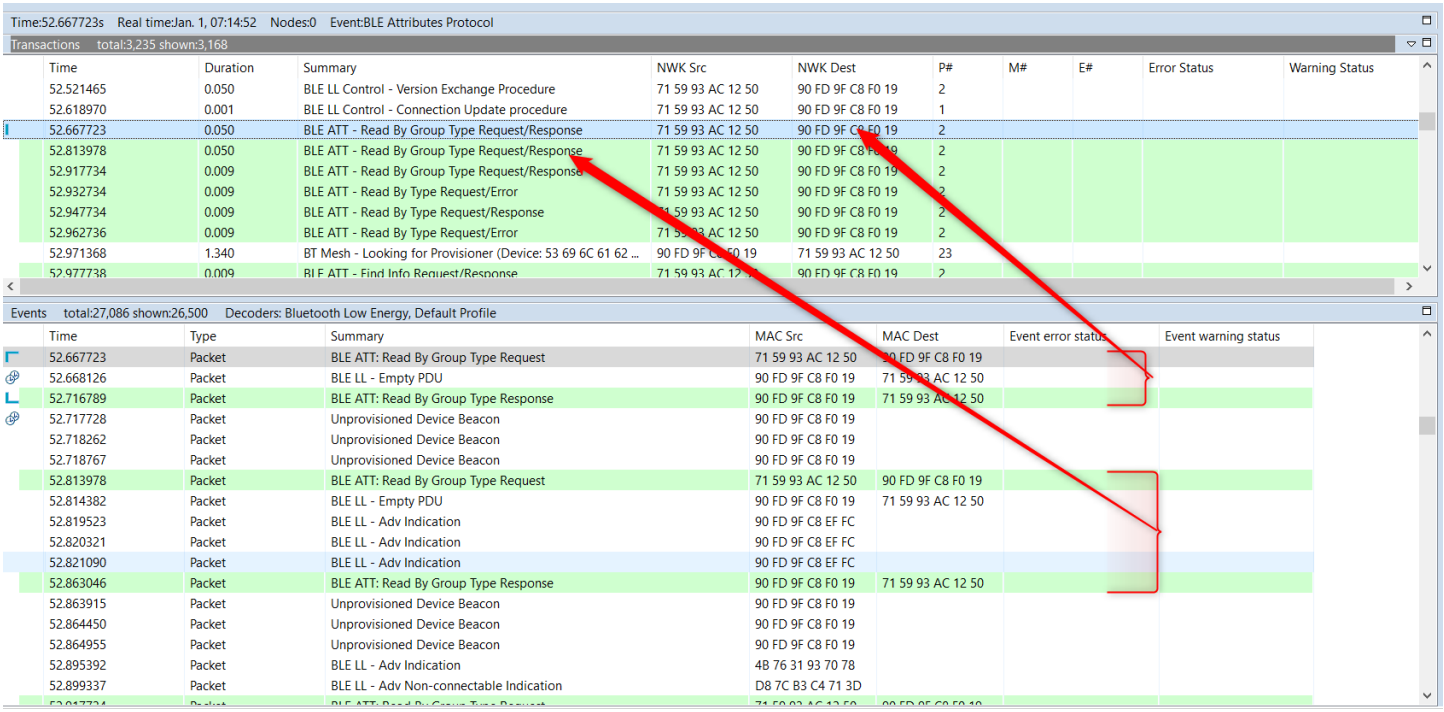
This section describes how the Network Analyzer can be used for Bluetooth Low Energy and Bluetooth mesh traffic monitoring. The current version of the Bluetooth core specification supported is 5.2. The current version of Bluetooth mesh profile and model specification is 1.0.1.

Bluetooth Low Energy profile support is limited. Additionally, there is no support for Bluetooth LE random addresses resolving.

4.1 Network Analyzer for Bluetooth LE

This section reviews the capabilities of Network Analyzer for the Bluetooth LE protocol.

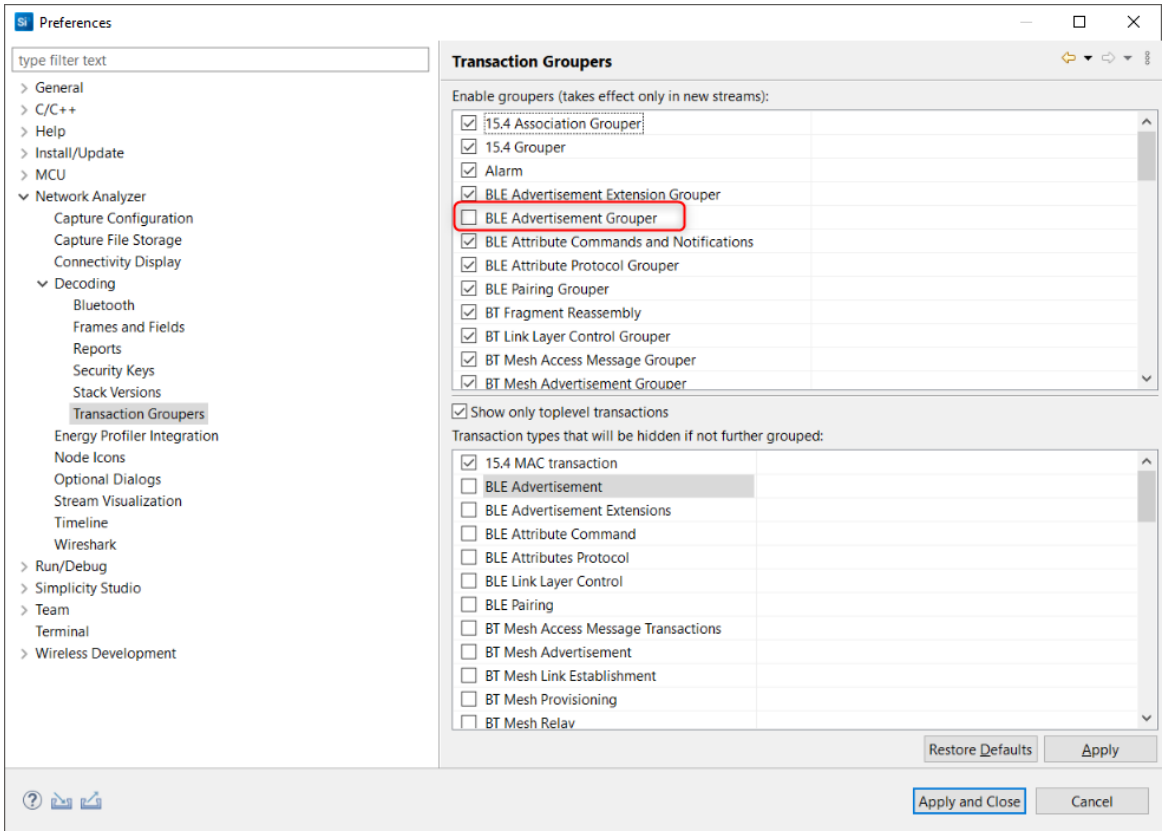
When Bluetooth Low Energy data is captured, Network Analyzer displays Bluetooth LE transactions and the corresponding events, as shown in the following figure.



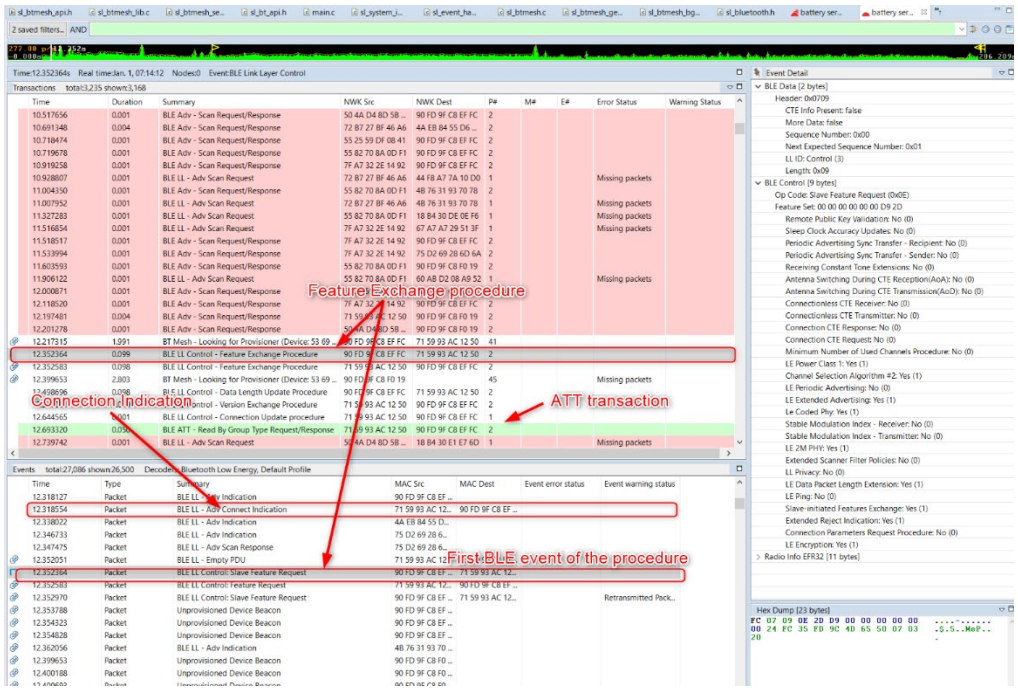
It shows that related packets like requests and responses together make a transaction. These transactions are listed separately in the Transactions pane. To find the first packet of the transaction, simply click on the transaction. To see the details of the packet, simply click on the packet. You can see both the raw and the parsed format of the packet in the Hex Dump / Event Detail pane (see section 3.2.2 Interval Editor for more detail).

To disable the display of some transactions that are not of interest, use **Preference > Network Analyzer > Decoding > Transaction Grouper**.

The following figure shows an example of “BLE Advertisement Grouper”.

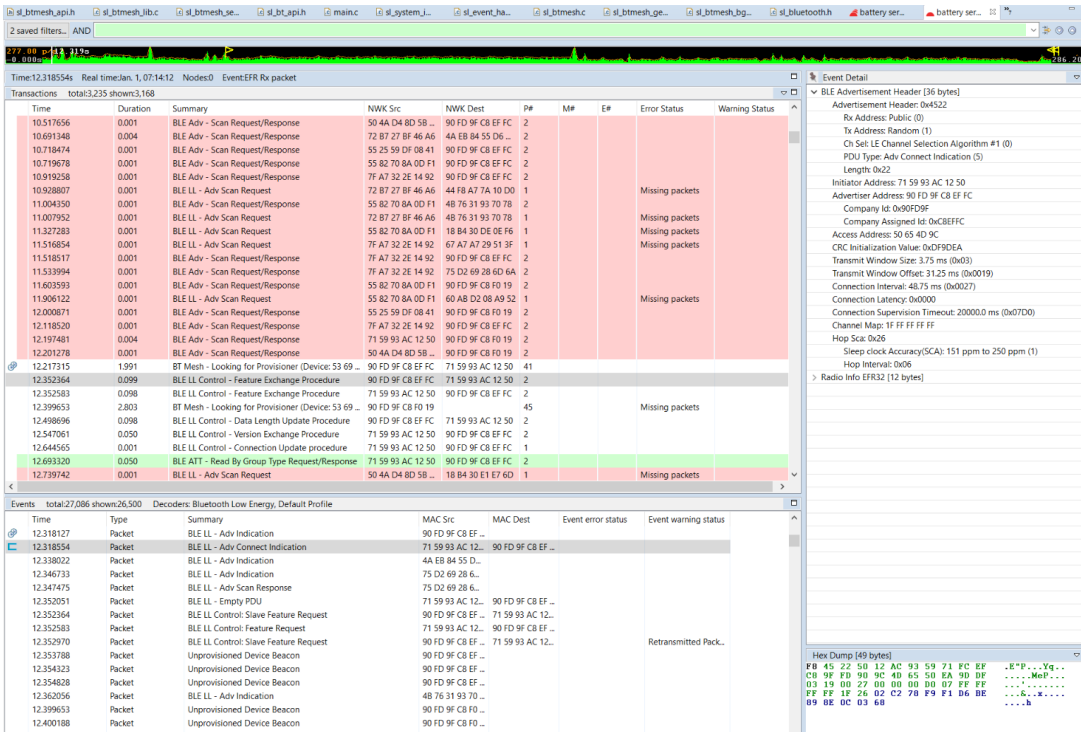


4.1.1 Bluetooth Low Energy Transaction Example



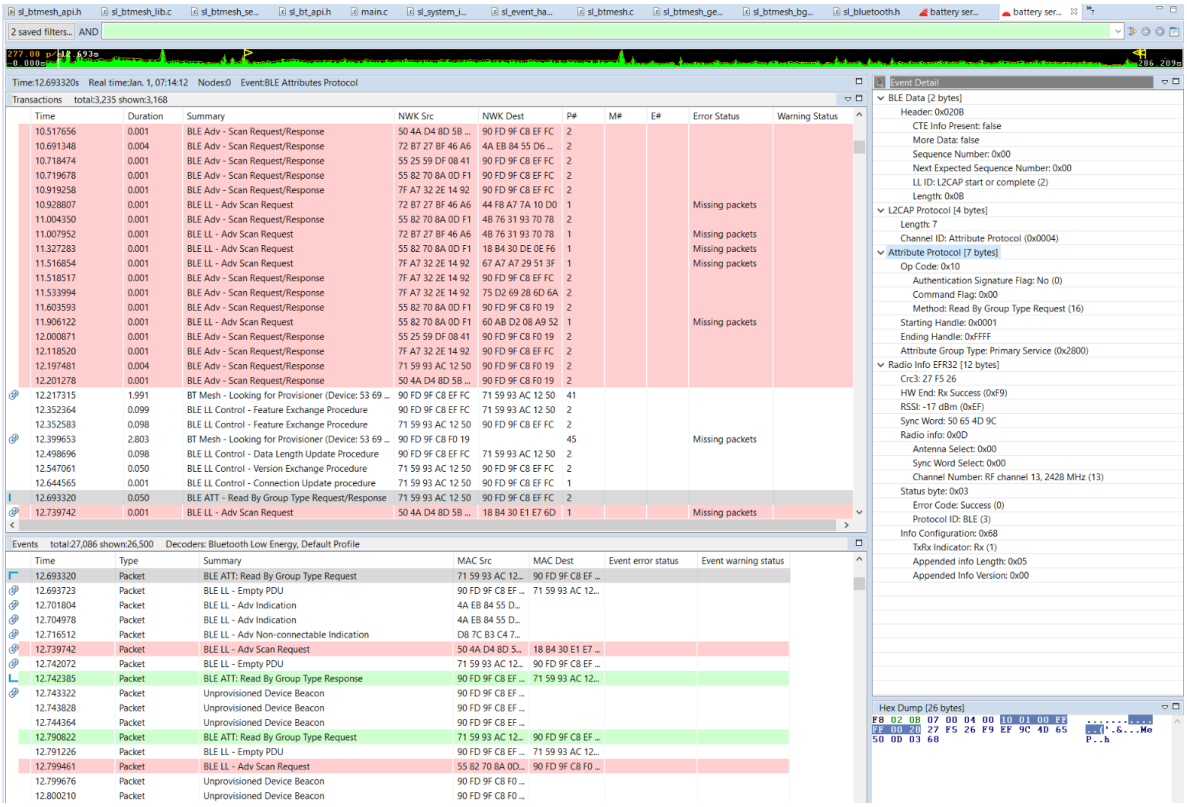
The Event Detail pane allows you to inspect packets at various levels, down to radio data. The example above shows a description of a Bluetooth Low Energy connection being established between a central and peripheral.

When the feature exchange transaction is selected, the Event panes display the corresponding Bluetooth LE events. Scroll up in the Event pane to find the Connection Indication packet.



The packets corresponding only to a particular Bluetooth LE connection can be filtered using the radio synchronization word. To do so, in the Radio info in the Event detail pane, right-click on “Sync word” and add to the filter.

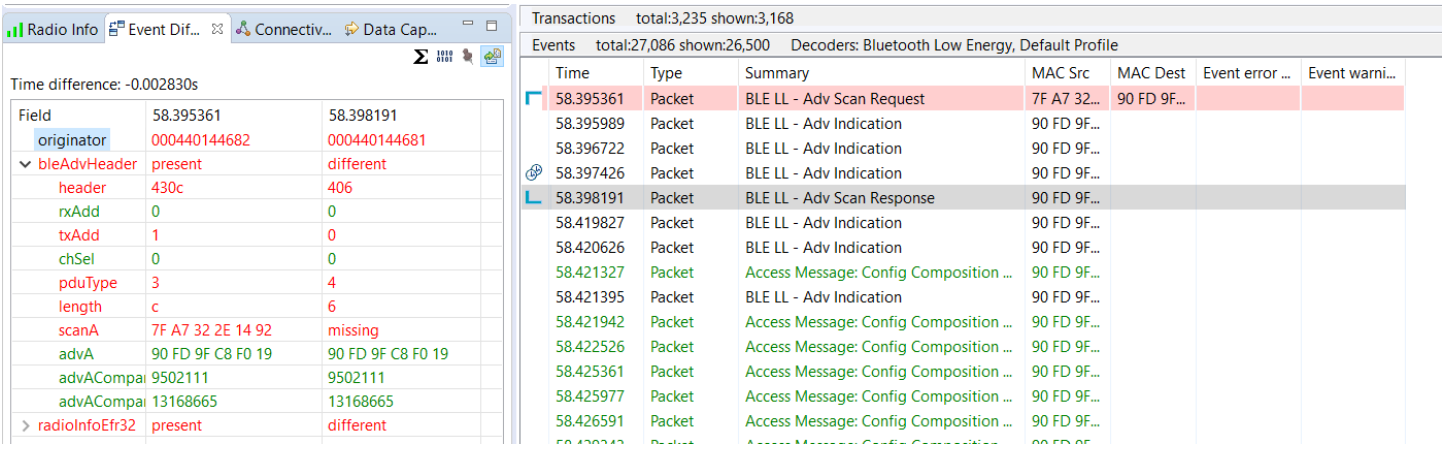
Once the Bluetooth LE connection is established, ATT transactions can take place. The following example illustrates a “Read by Group Type” ATT request and the corresponding response. Use the Event Detail pane on the right side to inspect the content of the request..



After a Bluetooth Low Energy connection is established, the next step is typically the GATT discovery of the GATT Client. A dedicated view is available for that under **Preferences > Network Analyzer > Decoding > Bluetooth,** which lists all GATT services/characteristics/descriptors and handles, can be used. There, the ability to save the view is also provided.

For more information on how Bluetooth Low Energy operates and how it can be monitored with Network Analyzer, refer to the Bluetooth LE connection and GATT connection flowcharts.

Additionally, the Event Difference pane can be used as a diffing tool between two different events. The following figure illustrates the event diff between a scan request and response.



4.1.2 Bluetooth Low Energy Data Decryption

Network Analyzer decrypts Legacy encryption automatically. In effect, the keys are harvested from the Bluetooth LE traffic data. When packets are encrypted using Secure Connection on the other hand, they can only be decrypted when using the security manager of the

Bluetooth Low Energy stack in debug mode. In the Silicon Labs Bluetooth Low Energy stack, this can be turned on using the following routine:

```
sl_status_t sl_bt_sm_set_debug_mode(void)
```

When using Secure Connections, this has the effect of having the Security Manager using debug keys. Those keys are also known by the Network Analyzer, which allows it to decrypt the Bluetooth LE data traffic.

To disable debug mode, restart the device. For more information, please refer to docs.silabs.com.

4.2 Network Analyzer for Bluetooth Mesh

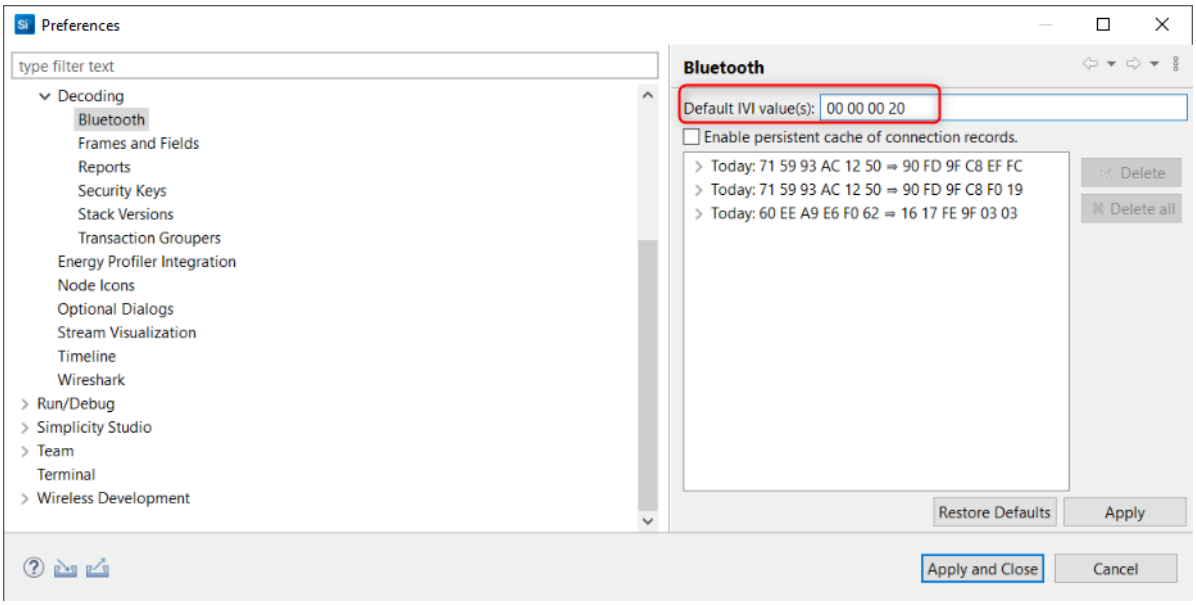
This section presents the capabilities of Network Analyzer for the Bluetooth mesh protocol. Network Analyzer offers the following features:

- Decryption of the Bluetooth mesh packets at all levels (network, application...)
- Handles Network-level segmentation / reassembly.
- Tracing packets from nodes out of RF reach from the PC (over Ethernet)
- Tracing packets from multiple nodes at once (several WSTKs connected over Ethernet)

As a reminder, Network Analyzer currently supports the Bluetooth mesh 1.0.1 profile and model specification. Support of Bluetooth mesh devices properties is limited. Provisioning Data PDUs cannot be decrypted and the map pane is not reliable.

4.2.1 Default IV Index Value

A Bluetooth mesh live or recorded session that has a non-zero IV index will not be decrypted properly. This can be adjusted in the Bluetooth decoder (**Preferences > Network Analyzer > Decoding** > **Bluetooth**) by setting the default IV index value. The following figure illustrates this.



Make sure to import the ISD capture file again to see the change being applied.

Note: More information on the IV index can be found in *AN1318: IV Update in a Bluetooth Mesh Network*.

4.2.2 Keys

The Bluetooth mesh stack is composed of several layers, starting from the network layer up to the access layer. Data traffic can be encrypted in various context (that is, stack layer levels):

- Network: Each Bluetooth mesh network has its associated network key. A node can have several network keys.
- Device: Each device in a particular network has its own device key.
- Application: Each application, depending on how it is configured, has its own application key.

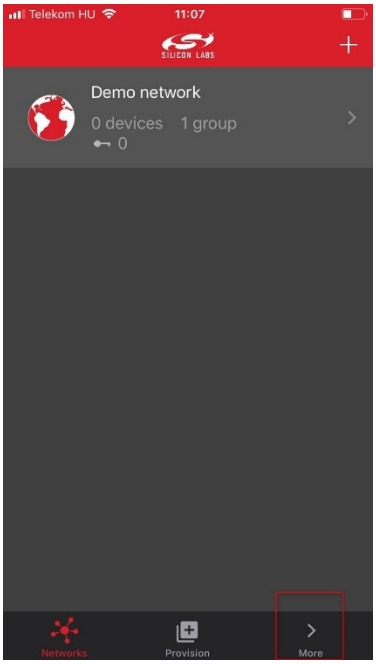
For more detail on how Bluetooth mesh security and encryption work, refer to the Bluetooth mesh profile specification.

When building a Bluetooth mesh network using a smart phone application or a gateway, it should be possible to export the corresponding security keys. The security keys can correspond to any of the three types: network, device, or application. The provisioner should allow you to export the keys in a text format that can then be shared with other applications.

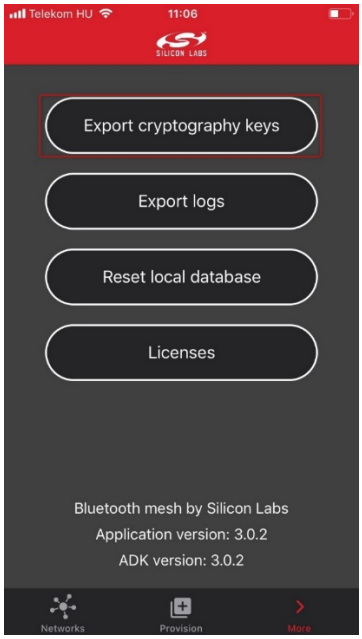
This is useful typically in the case of analyzing network traffic or rebuilding a network from scratch. Network Analyzer can import and export Bluetooth mesh keys. With the Silicon Labs Bluetooth mesh mobile application, a JSON text format is used for keys import/export.

The following steps indicates how to export keys using the Silicon Labs Bluetooth mesh phone application. Note that the steps are independent from the phone operating system, but the graphic layout of the smart phone application might differ.

1. Browse to the export menu.



2. Use the Export cryptographic keys button to create the corresponding JSON file. The JSON file called **MeshDictionary.json** can now be sent via email.

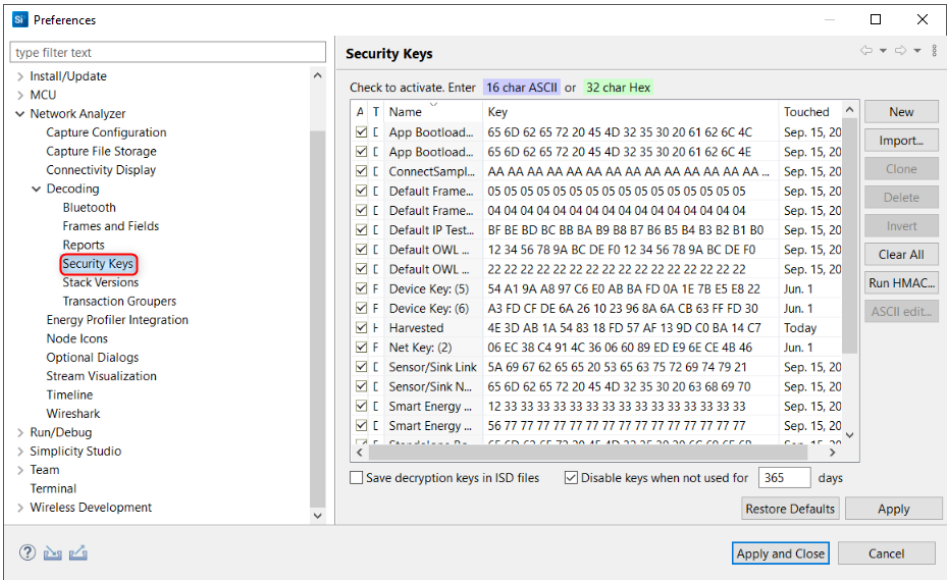


This should make a menu open allowing the user to select by which mean it wished to send the **MeshDictionary.json** file.

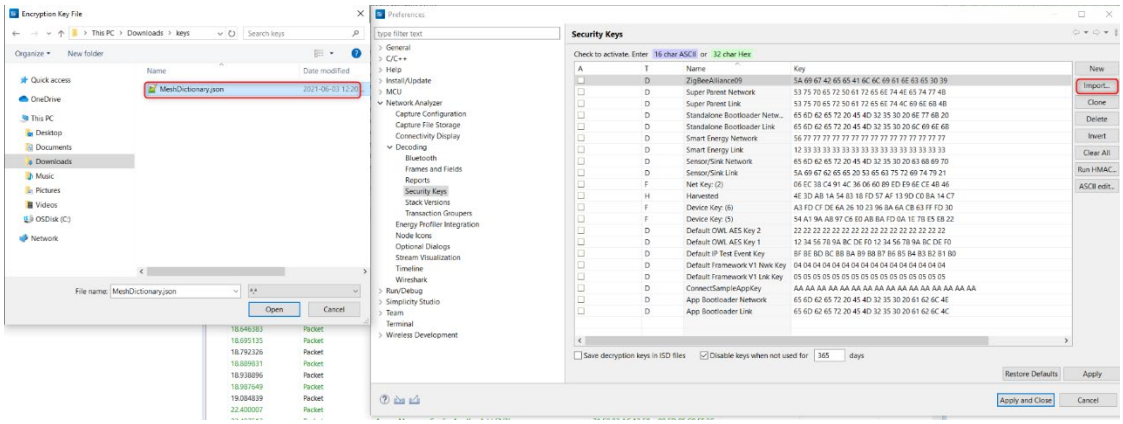
The content of the (generated) **MeshDictionary.json** JSON file is human readable text and contains a collection of keywords and hexa-decimal coded Bluetooth mesh keys.

The following steps indicates how to import the corresponding keys in Network Analyzer.

1. Go to **Preferences > Network Analyzer > Decoding > Security Keys**.

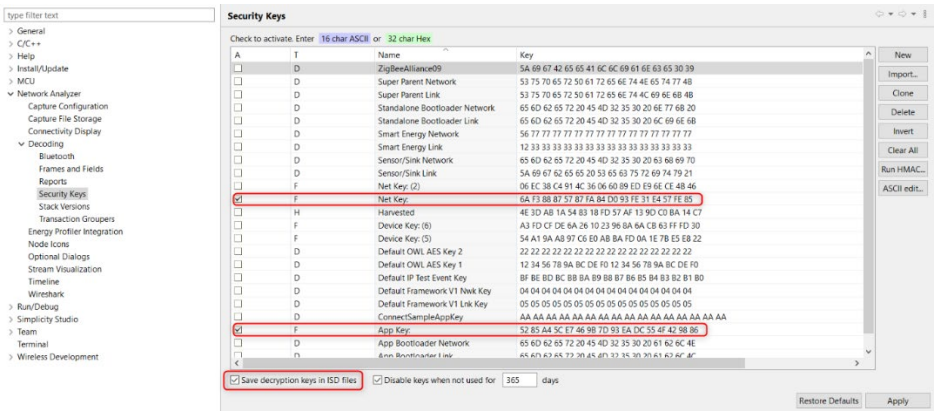


2. Deselect all default and saved keys that are already present and click **Import…**
3. Browse to the MeshDictionary.json file.



4. Click **Open** and see the keys imported in the “Security Keys“ window. For debugging purposes, the decryption keys can be saved in the ISD file (by checking the corresponding checkbox). This is not best practice from a security point of view but is acceptable for debugging. These instructions assume you have checked this.

5. Click **Apply and Close**.



6. After importing the keyfile, select **File > Other Network Analyzer Actions … > Reload** to refresh the data.
7. Finally, save the ISD file (checkbox in the bottom left of the security key dialog box).

The access layer Bluetooth mesh data should now be decrypted. The following sections show the expected result, in green on the top right corner of the figures.

4.2.3 Bluetooth Mesh Advertising Packets

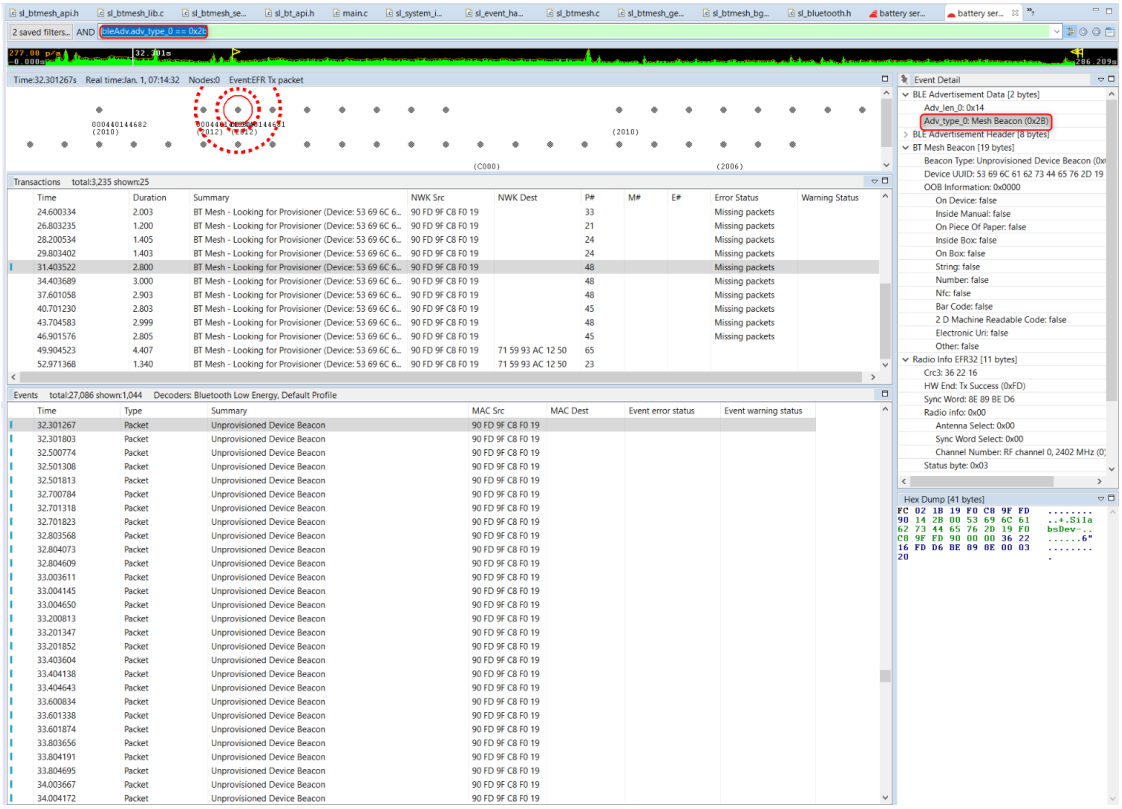
The Bluetooth mesh technology is based on Bluetooth LE advertising packets. Bluetooth mesh traffic is differentiated from the regular Bluetooth LE traffic through the AD types used by the Bluetooth mesh advertising packets. Network Analyzer can filter advertising packets in a number of ways.

The following AD types are used for advertising bearer-based Bluetooth mesh traffic:

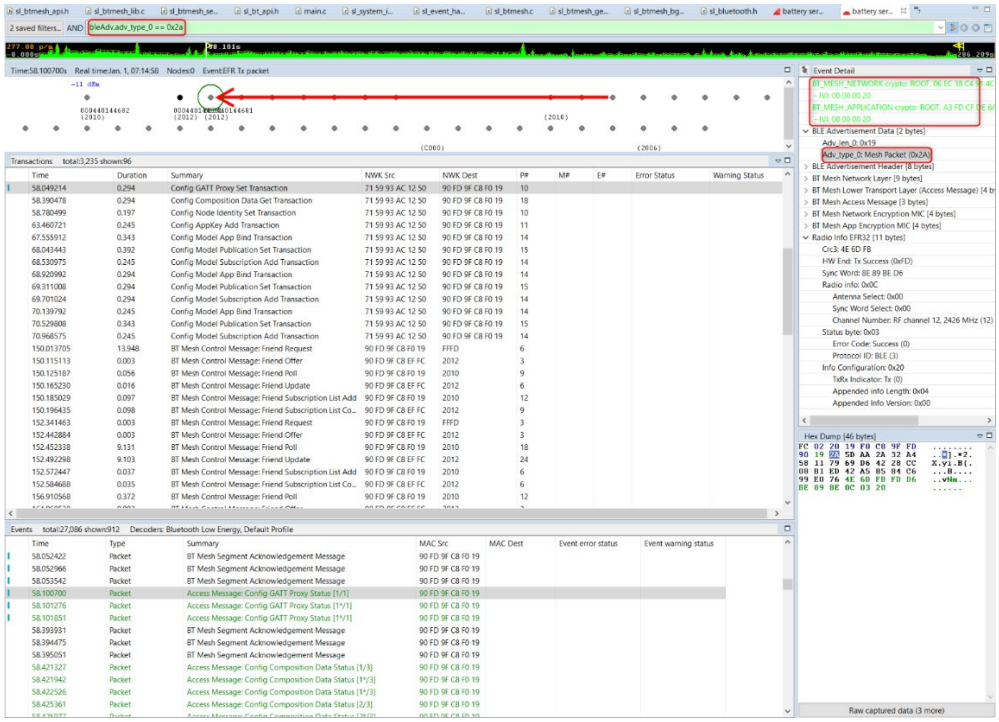
AD Type value	Data type name	Reference for definition
0x29	PD-ADV	Bluetooth Mesh Profile specification section 5.2.1
0x2A	Mesh Message	Bluetooth Mesh Profile specification section 3.3.1
0x2B	Mesh Beacon	Bluetooth Mesh Profile specification section 3.9

Based on this information, Bluetooth mesh advertising packets can be filtered. The following filters can be entered in the filter bar:

- `bleAdv.adv_type_0 == 0x2b`. This shows only Bluetooth mesh beacons that are used essentially for provisioning and secure data information propagation (secure beacons). The following shows an example.



- `bleAdv.adv_type_0 == 0x2a`. Shows Bluetooth mesh messages. Bluetooth mesh messages are used for common Bluetooth mesh data traffic. The payload of those mesh advertising packets is called “Network PDUs” (specified by the Bluetooth mesh profile specification). “Network PDUs” are the containers of the Network layer data. This is very useful because it allows you to display only the data traffic using the advertising bearer on provisioned nodes in a network. The following figure shows an example:



- `bleAdv.adv_type_0 == 0x29`. This shows Bluetooth mesh provisioning advertising packets. Those packets use the PB-ADV provisioning bearer and are used to provision a device using “Provisioning PDUs”.

4.2.4 Proxy Protocol

The Bluetooth mesh technology is mainly based on Bluetooth LE advertisement packets used along with Bluetooth mesh AD types. Nevertheless, in some cases, some devices are not able to advertise using the Bluetooth mesh AD types. As a consequence, the Bluetooth mesh specification allows communication over a GATT connection and uses what is called the Proxy protocol to exchange Network PDUs.

The proxy protocol is designed to enable nodes to send and receive Bluetooth mesh network packets over a connection-oriented bearer. As mentioned earlier, a node could support GATT but not be able to advertise the Bluetooth mesh Message AD Type. This node will establish a GATT connection with another node that supports the Bluetooth LE ATT bearer, called GATT bearer, and the advertising bearer, using the Proxy protocol to forward messages between these bearers.

Note: The term "GATT bearer", in effect, corresponds exactly to the ATT bearer as specified in the Bluetooth Host specification (Bluetooth Core specification, Host Vol. 3, 3.2.11).

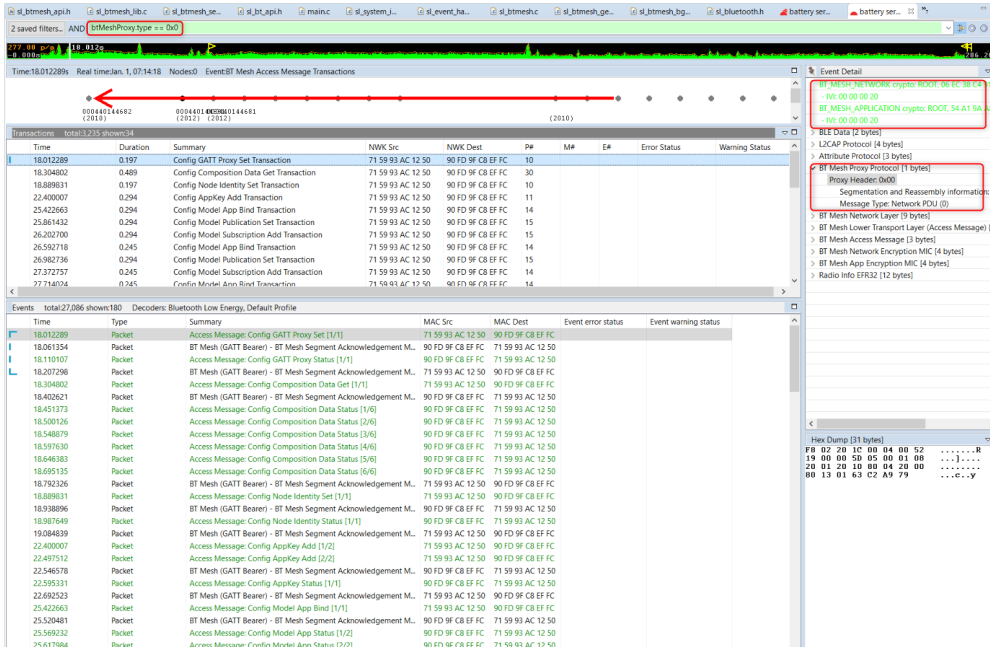
Once the Bluetooth LE connection is established, the node can send and receive what are called "Proxy PDUs". A Proxy PDU is essentially a data container for the following PDUs:

Type	Name	Description
0x00	Network PDU	The message is a Network PDU as defined in Section 3.4.4 of the profile spec.
0x01	Mesh Beacon	The message is a Bluetooth mesh beacon as defined in Section 3.9. of the profile spec.
0x02	Proxy Configuration	The message is a proxy configuration message as defined in Section 6.5. of the profile spec.
0x03	Provisioning PDU	The message is a Provisioning PDU as defined in Section 5.4.1. of the profile spec.

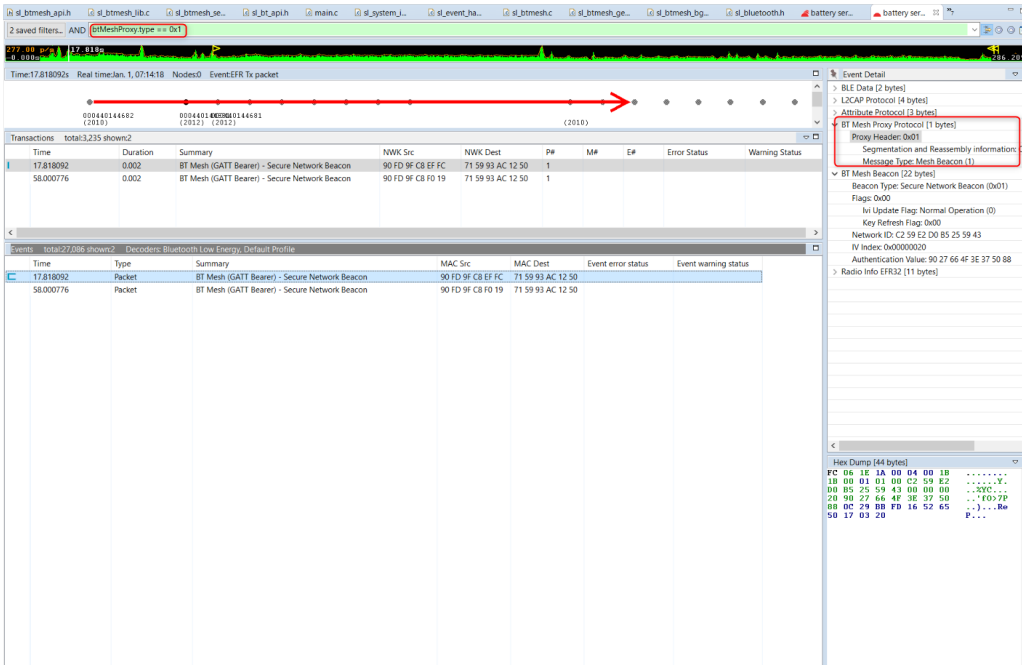
The Network PDU corresponds to all messages handled at the "Network Layer".

Network Analyzer allows you to filter Bluetooth mesh GATT bearer data in a live (or recorded) network session. Based on this information, Bluetooth mesh packets can be filtered. The following filters can be entered in the filter bar:

- `btMeshProxy.type == 0x0`. This shows all Network layer traffic. This is the GATT bearer equivalent to filter Bluetooth mesh Messages on advertising bearer traffic. The following figure shows an example (note the application and network key data in green in the top right corner).

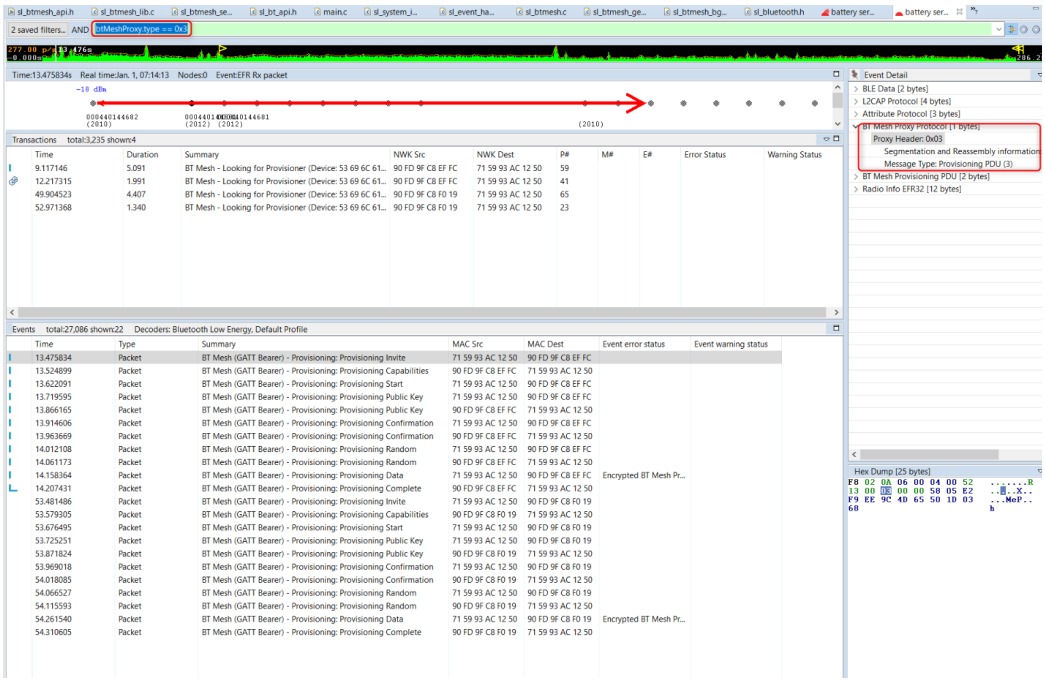


- `btMeshProxy.type == 0x1`. This shows Bluetooth mesh beacons on GATT bearer. The following figure shows an example, and shows an example of the secure beacon:



- `btMeshProxy.type == 0x2`. This shows Proxy client and server configuration messages. Proxy configuration messages are used to configure the proxy filters. The proxy server uses a filter to decide whether to forward the message to the proxy client or not. In practice that filter is only useful in certain specific cases.

- `btMeshProxy.type == 0x3`. This shows Provisioning PDUs over the GATT bearer. This is the GATT bearer equivalent of filtering PB-ADV adverting packet on advertising bearer Bluetooth mesh traffic. The following figure shows an example:



Note: When using Bluetooth mesh, the node map may not be totally reliable. This is a known issue in Network Analyzer.

Alternatively, filtering can be done on both bearers using frame patterns. When selecting a packet containing Bluetooth mesh Network layer data, in the Event detail pane, right-click the Bluetooth mesh Network data of that packet and click **Filter by Frame Pattern**. This filters all Network messages regardless of the bearer. The same procedure can be done with Bluetooth mesh beacons and provisioning PDUs.

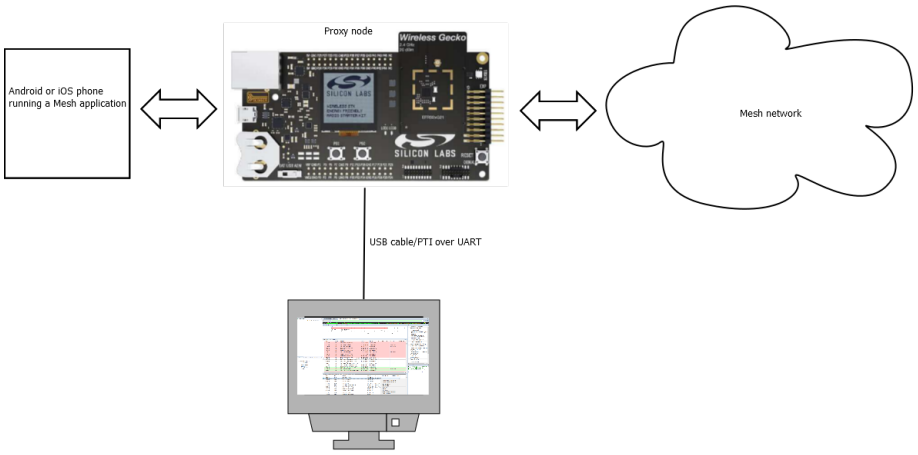
5 Bluetooth Mesh Networking and the Network Analyzer.

In a Bluetooth mesh network, especially on the field, node access can be challenging. Even when accessible, all nodes might not have the PTI pins exposed allowing network monitoring.

This section describes briefly what techniques can be used to monitor the Bluetooth mesh traffic in such environments.

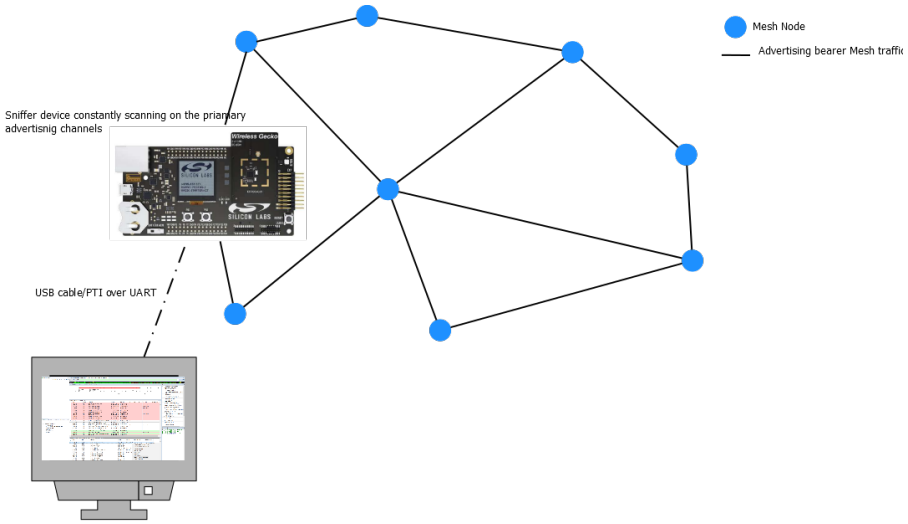
The Silicon Labs recommendation for this use case is to set up a WSTK-based proxy node. That would be, in effect, a dedicated sniffer node. The sniffer node does not need to run any particular model beyond the basics. It would only need to have the network key associated with the Bluetooth mesh network to decipher the traffic.

Using a mobile phone application, connect to the proxy sniffer node that would be used with the Network Analyzer to see the Bluetooth mesh traffic. That way, not only traffic on the advertising channels can be monitored but also data using the GATT bearer. This can be quite helpful when debugging issues that occur between the mobile phone application and the nodes.



Alternatively, a sniffer node can be created in a Bluetooth mesh network by simply loading a demo or sample application that is constantly scanning onto a WSTK-based Bluetooth LE device (i.e. an EFR32BG radio board mounted on a WSTK).

As an example, the soc-thermometer-host, constantly scanning for health thermometer server devices, would receive all Bluetooth mesh advertising-based PDUs. The fact that it is not a Bluetooth mesh node is not a problem as this can be recorded and decoded later on in Network Analyzer (see section 4.2 Network Analyzer for Bluetooth Mesh)



Although this type of sniffer is very useful in the field, it has a couple of limitations: GATT bearer Bluetooth mesh data would not be sniffed and only packets within radio range will be caught. In practice, the latter can be mitigated if the sniffer node is close to a relay node.

Finally, a standalone Java-based packet trace tool exist that is worth mentioning. The data stream recorded can be decoded in Network Analyzer (and Wireshark) or can simply be stored in text or binary format. For more detail, refer to Github.

Simplicity Studio

One-click access to MCU and wireless tools, documentation, software, source code libraries & more. Available for Windows, Mac and Linux!

IoT Portfolio
www.silabs.com/IoT

SW/HW
www.silabs.com/simplicity

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